Usually, Machine symbols are deleted to create a "user domain" with a subset of the configured NSX Machines.

If you delete a Machine from one map, but the Machine is represented in a second map, and if an alarm is generated by that Machine, then the alarm will only be represented on the map containing the Machine.

Please note that if you delete the Machine from *all* maps, then the machine object will be deleted from the database. If that Machine generates a new alarm, then a new database object will be created for that Machine, and the Machine will be represented in *all* maps again.

If you have an NSX Machine that sends alarms to the Director, but you don't want the NSX Machine to be represented in any maps, you should reconfigure the NSX machine to stop sending the alarms to the smid process running on the Director. This will prevent nrdirmap from receiving the events and creating the NSX Machine icon. If you still want the events from the NSX Machine to be logged in the Director machine, but not displayed by nrdirmap, configure the NSX Machine to send events to the loggerd daemon on the Director rather than the smid daemon.

If you want to "put back" symbols that you previously deleted, there are two things you can do if the objects that the symbols represented are still in the object database.

Option 1: You can use the **Edit→Add Object** menu function to add the object back to the map. Please note that you will get a few warning messages that you are adding an object that already exists. Press **OK** to continue adding the object. Please note that nrdirmap only creates symbols for alarms that are received *after* the machine/application are added to the map. If you want to add a Machine or Application that already exists in the database and you want to also see what alarms already exist, you should use Option 2.

Option 2: If you have many objects that you want to add back to the map, it might be faster to bring the user interface down, and then add the -f option to the nrdirmap registration file. This will force nrdirmap to represent all objects in the database on the map. Refer to the section in this chapter on editing registration files for more information on the -f option.

If one map is open (being viewed with ovw) and a second map is closed, and an event comes in, an alarm will be displayed on the open map. If the alarm is deleted from the open map before the second map is opened, then the alarm will be deleted from the database, and the alarm will not be represented on the second map when the second map is opened. On the other hand, if the second map is opened before the alarm is deleted from the first map, the alarm will be read from the database and will be represented in the second map.

Note that Object attributes apply to all maps. In other words, any time that the **Edit→Describe/Modify** option is used to edit an object, these changes will apply to all maps. Unfortunately, there is no way to customize Object attributes on a per-map basis.

Using Read-Only Maps

The NetRanger Director supports the use of read-only user interface sessions. Using read-only maps in OpenView can be a little tricky, so consider reading the appropriate OpenView documentation before using read-only maps.

Here are some things to remember about read-only maps:

- It is not possible to add symbols to or remove symbols from a read-only map. This is a restriction in OpenView that cannot be circumvented. This means that it is impossible for nrdirmap to create an alarm symbol on a read-only map.
- As a result, when an event is received by a read-only map, nrdirmap will change the status of the machine's application's "OkAlarm" symbol from Normal (green) to the status of the event that was received. This means that even on a read-only map, you should see the status color propagation when an alarm is received.
- Once you see the OkAlarm (and therefore, the Application and Machine) change color on a read-only map, you should use the Map→Refresh Map menu option to view an updated copy of the map. This will allow you to view the alarms that have come in.
- Please note that if you have no read-only copies of the map up, then refreshing the map will not provide you with additional information. You must have a read/write copy running to ensure that the map is updated properly.
- If a read-only nrdirmap receives an event, and if the application that generated the alarm does not have an "OkAlarm" displayed on that read-only map (for instance, if there are other alarms already displayed), then nrdirmap has no OkAlarm symbol to modify in order to reflect the new alarm. As a result, it is advised that users of read-only maps ensure that the maps contain OkAlarms so that status changes are seen immediately. If this is not possible, then read-only users should either refresh the maps occasionally or use *eventd* to notify the user that an event has been received and the map should be refreshed.

Limiting Access to NetRanger Director Security Information

There are functions in the NetRanger Director and in OpenView that can be used together to control which OpenView users have access to security management information.

If you have multiple users who have permissions to run the OpenView user interface, but if you want only a subset of those users to be able to view security information, read the section below.

Limiting access to security information is done by "disabling" the nrdirmap application for one or more OpenView maps. Once nrdirmap is disabled for a map, nrdirmap will not try to display security information on that map.

Once you have one or more maps that have nrdirmap disabled, and one or more maps that have nrdirmap enabled, you can set the permissions of the OpenView maps to limit which users can access which maps. The result is that only the users with permissions to open the nrdirmap-enabled maps will have the ability to view security information.

Disabling nrdirmap

nrdirmap is disabled on a per-map basis. You must decide whether or not to disable nrdirmap whenever you create a new map.

Note: Once a map is created and the decision has been made to enable

or disable nrdirmap, the decision is permanent. For instance,

once nrdirmap has been disabled for a map, it cannot be enabled.

Disabling nrdirmap can be done in two different ways, depending on how the map is created.

1. **If you create the map from the Map→Maps→New menu option, choose NetRanger Director from the list of configurable applications, and press the Configure button. You will see an option that reads**

   ```
   Should nrdirmap be enabled for this map?
   ```

   Choose `True` to enable nrdirmap and choose `False` to disable nrdirmap. The default is be `False` if the -d option appears in the nrdirmap registration file; the default is `True` otherwise.

2. **If you create the map from the command line using the ovw -m option, nrdirmap will be disabled if the -d option appears in the nrdirmap registration file; nrdirmap will be enabled otherwise.**

Setting User Groups

All users who should have access to security information MUST be in the group netrangr. Furthermore, all users who should not have access to security information should NOT be in the group netrangr.

On HP systems, users can be added to and removed from groups using the SAM utility. On Sun systems, the admintool can be used.

NOTE

On HP Systems, if a user is in the group "netrangr" (because of the configuration of the /etc/group file), but if the user's primary group is not "netrangr" (because the group ID listed in /etc/passwd for the user is not the group ID assigned to the group "netrangr"), then before the user can execute nrdirmap, the user must type the following command:

```
newgrp - netrangr
```

Sun users do not have to do this.

Setting Map Groups and Permissions

Once you have created your maps, you can use the `$OV_BIN/ovwchgrp` and `$OV_BIN/ovwchmod` commands to set map permissions.

Maps that have nrdirmap enabled should be readable by users who have access to the /usr/nr subdirectory. This means that maps that have nrdirmap enabled should be owned by the group "netrangr", and/or be owned by a user who is in the group "netrangr".

Ensure that map permissions are set so that the nrdirmap-enabled maps are not readable by people not in the netrangr group. For example, to allow only the user "netrangr" and users in the group "netrangr" to read and write a map called "default", type

```
$OV_BIN/ovwchown netrangr default
$OV_BIN/ovwchgrp netrangr default
$OV_BIN/ovwchmod 660 default
```

Use the `$OV_BIN/ovwls` command to list the maps and verify the permissions.

In the following example, two maps are created. The map called "secinfo" has nrdirmap enabled, and it will be accessible only by the user "angie" and by other users in the group "netrangr". The map called "nosecinfo" has nrdirmap disabled, and it will be accessible only by the user "bob" and users in the group "staff". Please note that the following example assumes that the -d option has not been added to the nrdirmap registration file.

Set-up

1. **Ensure that user "angie" is in the Unix group "netrangr".**
2. **Ensure that user "bob" is not in the Unix group "netrangr".**

Create the "secinfo" map

1. **Log in as user "angie".**
2. **Create the map by typing**

```
ovw -m secinfo &
```

Set the "secinfo" map permissions

1. **From the command line, type**

```
$OV_BIN/ovwchown angie secinfo
$OV_BIN/ovwchgrp netrangr secinfo
$OV_BIN/ovwchmod 660 secinfo
```

Create the "nosecinfo" map

1. **Choose the** Map→Maps→New **menu option.**
2. **Enter a map name, select "NetRanger/Director" from the list of configurable applications, and press the** Configure **button.**
3. **Choose** `False` **in response to the following question:**

```
Should nrdirmap be enabled for this map?
```

4. **Choose** OK **to create the map.**

Set the "nosecinfo" map permissions

1. **Type the following from the command line:**

```
$OV_BIN/ovwchown bob nosecinfo
$OV_BIN/ovwchgrp staff nosecinfo
$OV_BIN/ovwchmod 660 nosecinfo
```

Now, the map viewable by "bob" will not contain security information, but the map viewable by "angie" will.

Overview

nrConfigure is a Java-based graphical user interface (GUI) that allows you to remotely configure NetRanger applications and access those configurations. It supports all the functionality of the individual nrget, nrgetbulk, nrset, nrunset, and nrexec commands. This GUI-based environment allows you to see an application's tokens, each token's actions, and each action's optional values. Figure IV-11 illustrates the nrConfigure window.

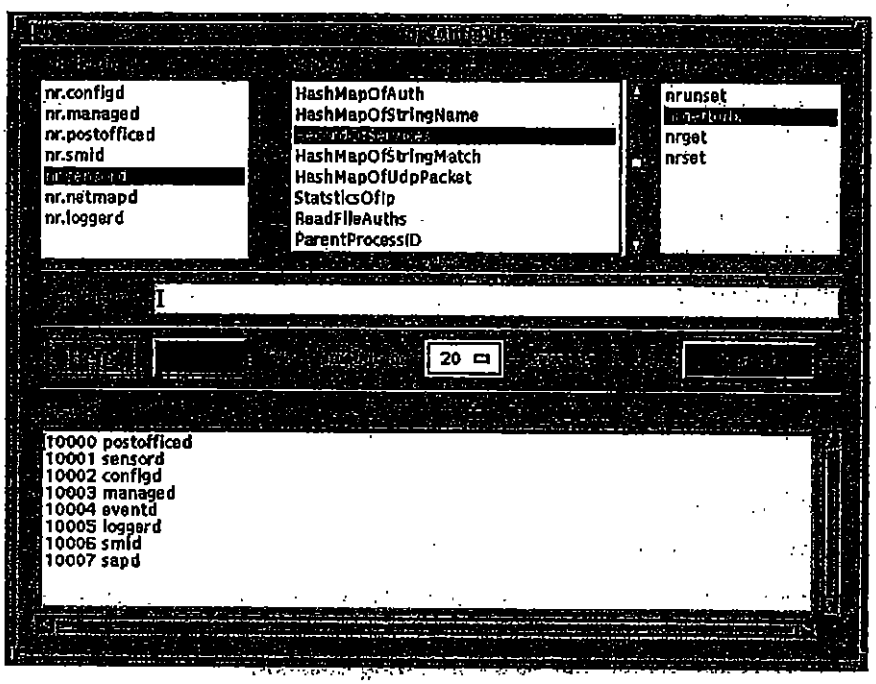


Figure IV-11: The nrConfigure Window

Architecture

The nrConfigure GUI works in the following manner. When you press **Execute** for a given application, token, and action, a packet is sent to *configd*, which sends it on to *postofficed*. *postofficed* sends it on to the proper daemon, which processes the requests and replies. The reply returns through *postofficed* to *configd* and appears in the nrConfigure's Results window.

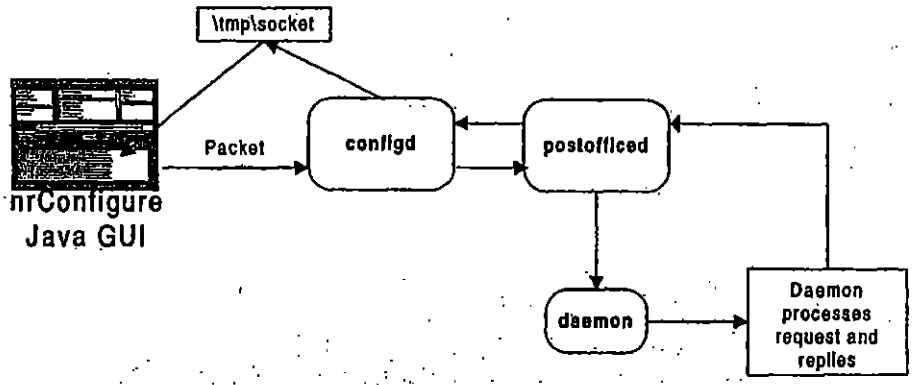


Figure IV-12: nrConfigure Architecture

Starting nrConfigure

nrConfigure is designed to configure daemon applications on an NSX or Director system. This is done by either selecting a machine icon from an OpenView security map or by invoking nrConfigure directly from the command line with the NSX or Director's organization and host id.

- **To start nrConfigure from within the Director, choose** Configure **from the Security menu.**
- **To start nrConfigure from the NSX command line, type the following command:**

```
/usr/nr/bin/nrConfigure <Organization ID>.<Host ID>:<Host Name> &
```

NOTE
The `<Organization ID>` and the `<Host ID>` can be found in /usr/nr/etc/hosts.
You may run multiple copies of nrConfigure at the same time.

Quitting nrConfigure

To quit any nrConfigure GUI, press the **Cancel** button. Pressing the **Cancel** button on one GUI will not quit any other nrConfigure GUI session.

Help with nrConfigure

If you need help with nrConfigure, follow these steps:

1. **Choose an Application from the nrConfigure GUI.**
2. **Press the Help Button.**

   A help window is displayed for the selected application. Help is also listed for all of the application-supported tokens.

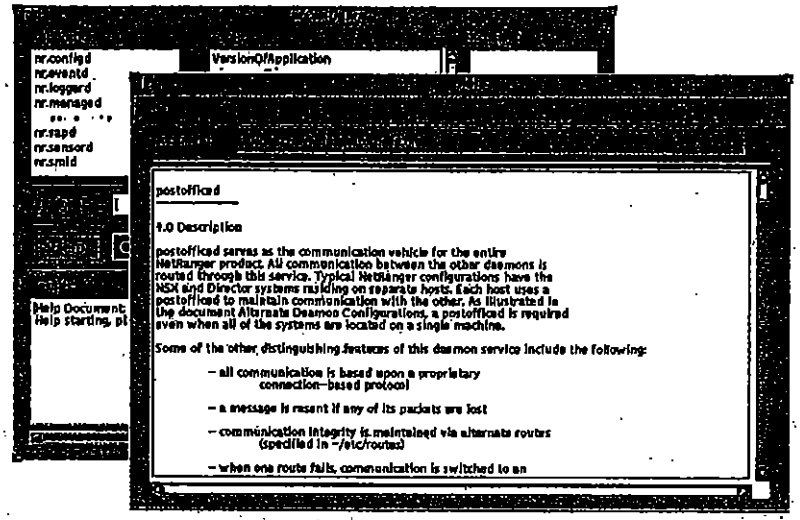


Figure IV-13: nrConfigure Help

Configuring an Application with nrConfigure

To configure an application with nrConfigure, follow these steps:

1. **Select the Application you wish to configure in the Application selection box.**

   The GUI will display that application's allowed Tokens.

2. **Select the Token you wish to execute in the Token Selection box.**

   The GUI will display that token's allowed actions.

3. **Select the Action you wish to execute in the Action selection box.**

   The GUI will display the following information:

   - Any Optional Parameters, if required
   - Any default values for Optional Parameters
   - Any set values for Optional Parameters

NOTE

The GUI will display Parameters without current values as the parameter's name in "<>" brackets.

4. **Press the Execute button.**

   The results will display in the Results field.

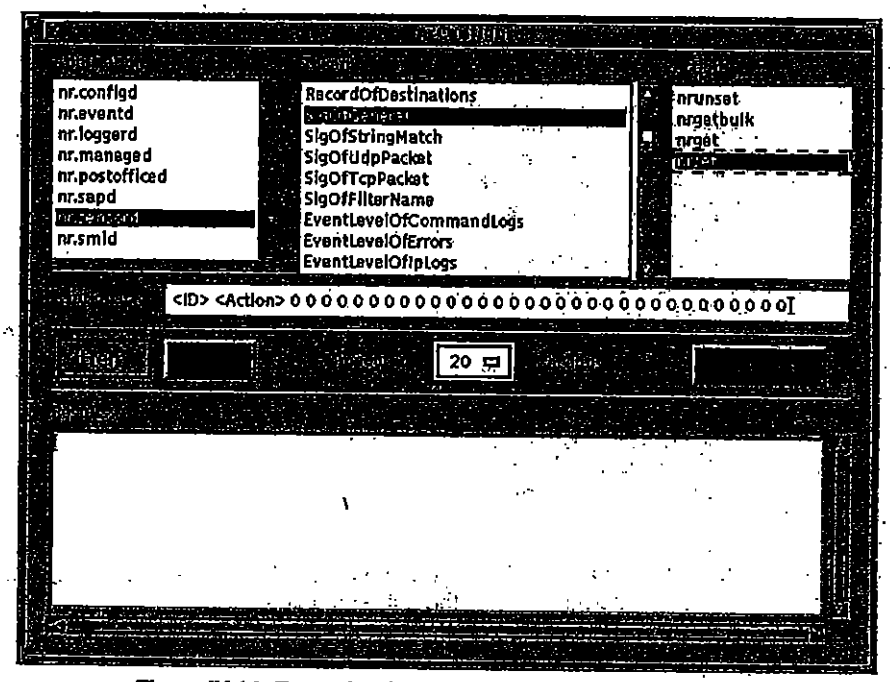


Figure IV-14: Example of Parameters in Optional Field

NetRanger Token Order in the nrConfigure GUI

The display of NetRanger tokens in nrConfigure is dependent upon the order or placement in the list of NetRanger tokens supported by a NetRanger Application or Daemon.

NetRanger Tokens supported by a NetRanger Application are specified in the application persistence file (/usr/nr/etc/persistence.applications). In the following sample, all the NetRanger Tokens are grouped by function (Filename, Read, Write).

```
APPLICATION=nr.postofficed
ID=10000
HELP=postofficed.txt
TOKEN=FilenameOfError
PARAMETER=filename
DEFAULT=../var/errors.postofficed
TOKEN=FilenameOfConfig
PARAMETER=filename
DEFAULT=../var/postofficed.conf
TOKEN=FilenameOfHosts
TOKEN=FilenameOfServices
TOKEN=FilenameOfAuths
TOKEN=FilenameOfDestinations
TOKEN=ReadFileConfig
TOKEN=ReadFileHosts
TOKEN=ReadFileServices
TOKEN=ReadFileAuths
TOKEN=ReadFileDestinations
TOKEN=WriteFileConfig
END
```

These same NetRanger Tokens could be grouped by the file they operate on (Config, Hosts, Services, Auths, and Destinations), as in the following sample.

```
APPLICATION=nr.postofficed
ID=10000
HELP=postofficed.txt
TOKEN=FilenameOfError
PARAMETER=filename
DEFAULT=../var/errors.postofficed
TOKEN=FilenameOfConfig
PARAMETER=filename
DEFAULT=../var/postofficed.conf
TOKEN=ReadFileConfig
TOKEN=WriteFileConfig
TOKEN=FilenameOfHosts
TOKEN=ReadFileHosts
TOKEN=FilenameOfServices
TOKEN=ReadFileServices
TOKEN=FilenameOfAuths
TOKEN=ReadFileAuths
TOKEN=FilenameOfDestinations
TOKEN=ReadFileDestinations
END
```


To edit the location of a NetRanger token in the list of Supported NetRanger tokens for a NetRanger Application, follow these rules:

1. **The "APPLICATION", "ID", and "HELP" header (HEADER) must come first to define a valid NetRanger Application.**

```
APPLICATION=nr.postofficed
ID=10000
HELP=postofficed.txt
```

2. **The "END" tag closes the definition (TRAILER).**

```
END
```

3. **Always move the trailing "PARAMETER" and "DEFAULT" tags, if they exist, for the selected token.**

```
TOKEN=FilenameOfError
PARAMETER=filename
DEFAULT=../var/errors.postofficed
```

4. **NetRanger tokens not defined between the HEADER and TRAILER will not show up in the nrConfigure NetRanger token list when the Application is highlighted.**

eventd

Overview

eventd is a daemon service shipped with the NetRanger Director package that allows users to receive notifications of alarm events via e-mail. *eventd* receives copies of alarms from *smid*, arranges them into a readable format, and e-mails them to users based on information stored in configuration files. An example e-mail follows:

```
From netrangr@director1.wheelgroup.com Tue Nov  5 12:01:10 1996
From: netrangr@director1.wheelgroup.com
Date: Tue, 5 Nov 1996 12:07:42 -0600
Subject: Alarm OUT->IN Level 2.

Date: 1996/11/05,12:06:00
NSX: [10001.1.100]
Attack: 8000,502 Level: 2
Addrs: 192.216.46.55:80->207.18.164.150:40134
Names: webcrawler.com->lookout.wheelgroup.com
Sig: 8000 "String Match"
Msg: "golf"
```

Basic Setup

eventd is not configured when the package is installed. To set up *eventd*, do the following:

- set up event script's configuration file
- set up eventd configuration file
- set up smid configuration file
- set up eventd to start

Set Up the Event Script's Configuration File

The event script converts alarms into e-mail messages and sends them to a configurable destination. *eventd* must be configured before it can send alarms. The default configuration file is a template with sample comments only. Unlike the configuration files for NetRanger's daemon services, the configuration files for the event *script* are located in /usr/nr/bin/eventd. Perform the following steps to configure the event script:

1. **Add your organization definition.**

   ```
   ORGANIZATION    100
   ```

2. **Add users to receive Type 2 alarms. Only one line for Type 2 alarms is accepted.**

   ```
   2       netrangr
   ```

3. **Add users to receive Type 3 alarms. Only one line for Type 3 alarms is accepted.**

   ```
   3       netrangr
   ```

4. **Add users to receive Type 4 alarms. Multiple lines may be added for Type 4 alarms. Type 4 alarms have 5 levels, a source, a destination, and recipients.**

   ```
   # Level 4 Alarms
   #
   # Source and Destination of Alarm
   # Src: OUT,IN,-
   # Dst: OUT,IN,-
   # "-" denotes either
   #
   #Type  #Level  #Src    #Dst    #Recipients
   4       1       OUT     IN      root
   4       2       OUT     IN      root
   4       3       OUT     IN      root
   4       4       -       -       user1
   4       4       OUT     IN      user2
   4       4       IN      OUT     user3
   4       4       -       IN      user4
   4       4       -       OUT     user5
   4       4       IN      -       user6
   4       4       OUT     -       user7
   4       5       -       -       root,user1,user2
   # Multiple lines are allowed for type 4 alarms.
   4       5       -       -       mcnealy@sun.com,bgates@ms.com
   ```

NOTE

Level denotes the level of the alarm you wish to select. Source and Destination have three possible settings: "IN," "OUT," and "-". IN denotes that the address must be inside the network. OUT denotes that the address must be from inside the network. The dash "-" symbol denotes that the address can be either. Source denotes the location of the IP address you wish to select for the recipients (as detailed above). Destination denotes the location of the IP address you wish to select for the recipients (as detailed above). Recipients denotes the mail destination of events that match the preceding selection process.

Set Up eventd's Configuration File

Before *eventd* can send alarm levels to destination, the following line in /usr/nr/etc/eventd.conf must be added:

```
EventApplication <UniqueID> <lowestLevelToSend> <locationOfScriptRelativeTo/usr/nr/bin>
```

The following example defines one script for one organization:

```
EventApplication 1002 2 ./eventd/event
```

Set Up smid's Configuration File

In order for eventd to receive alarm events from *smid* the configuration file smid.conf must contain a `DupDestination` entry for *eventd*. An example entry might look as follows:

```
DupDestination director1.wheelgroup eventd  2 ERRORS,COMMANDS,EVENTS
```

Set Up eventd to Start

Edit /usr/nr/etc/daemons by uncommenting the following line:

```
# nr.eventd
```

NOTE

If the above line is uncommented, then *eventd* will not be able to start after a reboot.

Advanced Setup

Monitoring Multiple Organizations.

If you are monitoring only one (1) organization, use event.conf as your configuration file. Otherwise, build a separate copy of event and event.conf for each organization. To do this, add links pointing to the original event under the name of the organization and make a copy of event.conf under the same name as the new event script. To edit each organization-specific configuration file, follow these steps.

1. **Create a script file with the same name as the configuration file. Use UNIX link command:**

   ```
   %cd /usr/nr/bin/eventd

   %ln -s ./event ./event_wheelgroup
   ```

   The script file, when run, will look for a configuration file by the same name with ".conf" appended. Use event.conf as an example.

2. **Make a copy and edit it according to your local needs:**

   ```
   %cp event.conf event_wheelgroup.conf

   %vi event_wheelgroup.conf
   ```

3. **Edit /usr/nr/etc/eventd.conf by adding organization designations, for example:**

   ```
   EventApplication 1002 2 ./eventd/event_wheelgroup

   EventApplication 1003 2 ./eventd/event_organization2
   ```

   In the example above, two scripts for two organizations are being added. The organizations are designated as wheelgroup and organization2.

For example, a new script for wheelgroup is made under the name of event_wheelgroup, and event_wheelgroup.conf, respectively.

The lines look this before configuration:

```
-rwxr-x---   1 netrangr netrangr    6603 Nov 22 17:19 event
-rwxr-x---   1 netrangr netrangr    1021 Nov 22 15:30
event.conf
```

and like this after configuration:

```
-rwxr-x---   1 netrangr netrangr   6603 Nov 22 17:19 event
-rwxr-x---   1 netrangr netrangr   1021 Nov 22 15:30 event.conf
lrwxr-xr-x   1 netrangr netrangr      5 Nov 22 15:31 event_wheelgroup -> event
-rwxr-x---   1 netrangr netrangr   1305 Nov 23 12:47 event_wheelgroup.conf
lrwxr-xr-x   1 netrangr netrangr      5 Nov 22 15:31 event_organization2 ->
event
-rwxr-x---   1 netrangr netrangr   1305 Nov 23 12:47 event_organization2.conf
```

Changing the Event Error Notification user

Event Error Notification is in the event script. It looks for its own configuration file, and if it cannot find it, or it has an error processing the alarm, it notifies the person(s) named as MAIL_FAILURE. To change user to receive e-mail on problems with the configuration file, edit eventd in the following manner:

```
%vi event
MAIL_FAILURE="userToReceiveEmailOnFailure@somehost.com,netrangr
"
```

For example:

```
MAIL_FAILURE="bob@alarmsRus.com,postmaster@walrus"
```

In the above example, "postmaster@walrus" is the second user added to the MAIL_FAILURE line.

Event Signatures

Event signatures consist of a unique signature identifier (SigID) and a sub-signature identifier (SubID). This section includes a list of the event signatures currently defined in NetRanger. SigIDs are built into NetRanger; SubIDs change depending upon the SigID they come under.

TCP/IP Event Signatures

TCP/IP event signatures are currently divided into three groups: **context-, content-, regular expression-** and **NetSentry-based**.

Context-Based Signatures

Context-based signatures are based on information passed in the TCP/IP header. This can include such things as the destination port, i.e. TCP port 80 for WWW traffic, IP Options, i.e. source routing, or a combination of events found in a hacking attack. The following group of signatures are context-based attacks that are detectable using NetRanger. signatures are generated from stateful multi-header packet analysis. NetRanger detects stateful attacks by remembering a sequence of events, or by reconstructing packets to find certain strings or attacks. Those attacks for which alarming is dependent upon maintaining the state of a connection are indicated by a "" following the attack name. These signatures currently analyze the following types of events:

- **Source Routing**—NetRanger detects both loose and strict source routing which is commonly used by hackers to bypass rules found in filtering routers.
- **ICMP Network Sweeps**—This attack uses the ICMP protocol to discover which machines are alive on a remote network. This is most often used as the first step of an attack to find potential targets. This can be implemented three different ways using different ICMP types. All three are detected within NetRanger
- **Fragmented ICMP traffic**—To get around filtering on large ICMP traffic, it is possible to fragment this traffic and "trick" some intrusion detection systems. NetRanger checks for and alarms on this activity.
- **Large ICMP traffic** —Numerous computers are vulnerable to an attack where if you send an ICMP packet with an extremely large data size it will crash the machine. NetRanger blocks and alarms this traffic.
- **TCP Port Sweep**—When targeting a specific machine, a hacker will frequently run a TCP port sweep to get a list of all available services on the remote target.
- **Half Open SYN Attack** —This attack was recently publicized when it was used to shut down several Internet Service Providers. This attack can crash a machine by overloading it with TCP connection requests that it never closes.
- **TCP Hijacking** —This attack looks for a characteristic of attacks which take over an existing TCP connection.

- **UDP Port Scan**—When targeting a specific machine, a hacker will frequently run a UDP port sweep to get a list of all available services on the remote target.
- **SATAN Scan**—This looks for both the normal and heavy SATAN attacks.

Content Based Signatures

For these attacks, NetRanger looks further than simple TCP/IP header information. It actually looks inside the packet for data which indicates an attack in progress. Most of these signatures take advantage of looking for these signatures within a certain context. For example, Sendmail attacks look for certain strings only within the sendmail port 25.

- **Smail attack**—NetRanger looks for an attack which was only found in the e-mail package "smail".
- **Sendmail invalid recipient**—This looks for attacks which try to send an e-mail to a program on a remote machine. The attack expects the remote machine to execute the e-mail as if it were a program.
- **Sendmail invalid sender**—This attack is like the previous attack except that the sender of the e-mail appears to be a program. When an error occurs within the e-mail the remote machine attempts to return it to the original sender. The e-mail is then executed as a program on the remote machine.
- **Sendmail reconnaissance**—This attack is used to gather information about remote users through the mail port. This is usually used as a preface to other attacks.
- **TFTP password**—This looks for anyone attempting to get the password file using the TFTP service which requires no authentication.
- **DNS HINFO Requests**—This attack uses DNS to gather information about a specific host.
- **DNS Zone Transfer Request**—This attack attempts to gather information about all hosts registered with your DNS server. This can be used by hackers to try to get a map of your network.
- **DNS request for all records**—This attack requests all records maintained on a remote server and it is used to gather information for a future attack.
- **RPC port registration**—This is used to register a specific application to a port on a remote machine and should never occur across the network.
- **RPC port unregistration**—This is used to unregister a specific application to a port on a remote machine and should never occur across the network.
- **RPC dump**—This is used to query a remote machine about which services are running at what ports. This is commonly used by hackers as a preface to an attack.
- **Proxied RPC request**—This is an attack where you trick the remote machine into issuing a request to a service such as NFS for you. This makes the source address appear to be the local machine instead of the attacking machine.

- **NFS mountd request**— This is an attack where a remote user tries to connect to the mountd process. This can be used to determine what file systems a site is sharing on the network and how it might be exploited.
- **Rexd request**—This is a request to the remote execution daemon and is used to run programs on a remote machine without authentication.
- **YP attacks**—There are five separate attacks which NetRanger looks for which try to take advantage of the service which maintains network password and other files.
- **Loadmodule attack**—NetRanger looks for a string characteristic of an attack which tricks a set uid program into giving system administrator privileges to the attacker.
- **Any matched string**—NetRanger can look for any string within any service using regular expression matching. This allows any organization to define their own requirements and look for abuse of their proprietary systems.

IP Options Events

IP options consist of a variable-length list of optional information for an IP datagram. Options are rarely used and not all routers and host support them. A good security measure is to refuse routing IP datagrams with options. Detecting IP options from externally connected networks provides a good indicator of potential network problems and attacks.

1000 IP options—Bad option list

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 1

The list of IP options for the datagram is incomplete or malformed. This may indicate an attack where the attacker is using improperly developed hacking software.

1001 IP options—Record packet route

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 2

This option instructs each router to record in the options list the IP address of the interface that the packet will be transmitted from. Because of the limited size of an IP header, only nine addresses can be store within this list.

1002 IP options—Timestamp

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 1

This option is similar to the record route option. Routers are requested to add timestamps into the options list. This option is of little value because of the limited size of the options field.

1003 IP options—Provide s,c,h,tcc

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 1

Basic security options as defined in RFC 1038. Used to carry security level and accrediting authority flags. Implementation of security via these options is obsolete and should not be used.

1004 IP options—Loose source route

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 5

This option specifies a list of IP addresses that the IP datagram must traverse (excluding the routers the datagram can also pass through). An attacker may transmit datagrams into a network using spoofed source addresses that appear to come from the target network. Responses to these packets are transmitted back to the attacker because the host recognizes the source route option. This allows attackers to defeat IP address based authentication mechanisms. Source route attacks usually use *loose routing* due to the number of hops a datagram must traverse across the Internet.

1005 IP options—SATNET id

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 1

Stream identifier option. This option is obsolete and should not be encountered.

1006 IP options—Strict source route

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 5

This option specifies the exact list of IP addresses that the IP datagram must traverse. This is similar to loose source routing. Source route attacks usually do not use this type of source routing.

ICMP Events

ICMP datagrams are generated to provide administrative and diagnostic information. The most common use of ICMP datagrams is the "ping" program, which verifies the existence of a host. ICMP traffic can provide much information about a network, and it can also modify network characteristics such as routing tables. Attackers use ICMP to discover and modify this information. SubIDs for the following signatures are set to zero.

2000 ICMP Echo Reply

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 1

This message is generated in response to an echo request. This type of datagram is sent from the host being 'pinged'.

2001 ICMP Unreachable

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 1

This message is transmitted to a host stating that the intended destination is unreachable for the IP datagram it transmitted. The first 64 bits of the failed datagram is transmitted along with the unreachable message. Unreachable messages can be used to disrupt existing TCP sessions.

2002 ICMP Source Quench

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 2

This message is transmitted to a host to report network congestion and requests a reduction in the current rate of datagram transmission. While not all systems support this feature, it can be used to enact a denial-of-service attack. The performance of the targeted host can be compromised by sending a continuous stream of these source quench messages.

2003 ICMP Redirect (change a route)

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 1

This message is transmitted from a router to host with an update for the host's routing table. This enables hosts to start with the minimal routing configuration that is subsequently updated by the network's router tables. Attacker's use redirect messages to place incorrect routes into a target host's routing table to support IP hijacking and other attacks.

2004 ICMP Echo Request

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 1

This message requests that the destination host transmit back an echo reply message. This type of datagram is sent to the host being 'pinged'. Individual requests are not a security threat. Large numbers of these requests may be a denial of service attack or network reconnaissance as depicted in SigID 2100 below.

2005 ICMP Time Exceeded for a Datagram

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 2

This message was designed to report circular or excessively long routes. A router decrements the time-to-live (TTL) counter whenever it processes a datagram and discards the datagram when the count reaches zero. The first 64 bits of the discarded datagram are transmitted along with the time exceeded message to the originating host. These messages typically occur when an attacker is using the "traceroute" program to help map out a target network. It also occurs when a host has the default TTL set to low (i.e. 30) and transmits datagrams to a destination far across the Internet.

2006 ICMP Parameter Problem on Datagram

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 1

This message is transmitted when a datagram header is incorrect. The first 64 bits of the incorrect header is also sent.

2007 ICMP Timestamp Request

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 2

This message requests the destination host to transmit back a timestamp reply containing the host's current time. Individual requests are not a security threat. Large numbers of these requests may be a denial of service attack or network reconnaissance as depicted in SigID 2101 below.

2008 ICMP Timestamp Reply

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 1

The message generated in response to a timestamp request. Use of this feature could be considered as another type of "ping".

2009 ICMP Information Request (obsolete)

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 2

This type of ICMP packet is obsolete and should not be used.

2010 ICMP Information Reply (obsolete)

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 1

This type of ICMP packet is obsolete and should not be used.

2011 ICMP Address Mask Request

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 2

This message asks the destination host to transmit back the address mask in use on the network. This service was designed to allow diskless clients to set the address mask by broadcasting this request over the local network. Individual requests are not a security threat. Large numbers of these requests may be a denial of service attack or network reconnaissance as depicted in SigID 2102 below.

2012 ICMP Address Mask Reply

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 1

The message generated in response to an address mask request. RFC 950 added this feature, but RFC 1122 forbids a host from sending replies unless it has been explicitly configured as an authoritative agent for address masks. Hosts that do not implement this feature correctly may respond to a targeted request - another type of "ping".

2100 ICMP network sweep w/Echo

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 5

This signature identifies a host that is transmitting ICMP echo request datagrams to multiple hosts on the network. This method is commonly used by attackers to identify active hosts within a network address range. While this can be a serious attack, network management tools such as HP OpenView also perform this type of network discovery on a regular basis.

2101 ICMP network sweep w/Timestamp

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 5

This signature identifies an attacking host that has transmitted ICMP timestamp request datagrams to multiple hosts on the network. While this request can also be used to identify active hosts within a network address range, most attackers use the more common echo request method (see SigID 2100).

2102 ICMP network sweep w/Address Mask

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 5

This signature identifies an attacking host that has transmitted ICMP address mask request datagrams to multiple hosts on the network. While this request can also be used to identify active hosts within a network address range, most attackers use the more common echo request method (see SigID 2100).

2150 Fragmented ICMP Traffic

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 4

This signature identifies an attacking host that has transmitted a fragmented ICMP packet. By design, ICMP packets are small and should never be fragmented. There are attacks that utilize fragmented ICMP packets to crash target systems.

2151 Large ICMP Traffic

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 4

This signature identifies an attacking host that has transmitted a large ICMP packet. By design, ICMP packets are small and should never be fragmented. There are attacks that utilize large ICMP packets to crash target systems.

3000 TCP Connection Logging

SubID Values: 0 - 65536 [SYN] Destination TCP port

100000 - 165536 [SYN-ACK] Destination TCP port + 100000

200000 - 265536 [FIN] Destination TCP port + 200000

300000 - 365536 [RST] Destination TCP port + 300000

Misc Field Info: TCP sequence number

Recommended Alarm Value: 1

Packets Required: TCP packets with the SYN, FIN, or RST flags set

This is event is used for logging TCP traffic. The token ***LevelOfTrafficLogging*** is used to configure the level of logging.

Level 1: No TCP logging occurs.

Level 2: Only TCP SYN packets are logged (default). This indicates that the source host has initiated an attempt to establish a TCP connection to the destination host using the TCP ports specified. The SubID for this signature is the destination TCP port. If the destination host refuses this connection because the requested port does not exist, an ICMP unreachable message will immediately follow.

Level 3: All TCP SYN, FIN, and RST packets are logged.

The sequence numbers stored in the miscellaneous field provide enough information to determine the number of bytes transferred within a TCP connection. This is accomplished by subtracting the initial sequence number from the SYN packet from the final sequence number from the corresponding FIN packet.

3001 TCP port sweep

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 5

This signature identifies an attacking host that has initiated a series of TCP connections to a number of different destination ports on the target host. This method is used by attackers to determine the services available on the target host for potential exploitation.

3050 Half-open SYN attack

SubID Values: 0

Misc Field Info: Affected TCP port on target system

Recommended Alarm Value: 5

This signature identifies the targeted host and TCP port where the source address and port may be randomly generated by an attacker. Detection of this signature is currently limited to FTP, Telnet, WWW, and E-mail servers.

3100 Smail attack

SubID Values: 0

Misc Field Info: "to: bounce"

Recommended Alarm Value: 4

This signature detects the very common "smail" attack against e-mail servers.

3101 Sendmail Invalid Recipient

SubID Values: 0

Misc Field Info: "to: |"

Recommended Alarm Value: 4

This signature detects any mail message that is transmitted to an address of "pipe" something. Due to the vulnerabilities previously discovered in sendmail and the complexity of the software, destination addresses of this type should not be allowed.

3102 Sendmail Invalid Sender

SubID Values: 0

Misc Field Info: "from: |"

Recommended Alarm Value: 4

This signature detects any mail message that is transmitted with a return address of "pipe" something. This should not be allowed under any circumstance.

3103 Sendmail Reconnaissance

SubID Values: 0

Misc Field Info: "vrfy" or "expn"

Recommended Alarm Value: 2

This represents a reconnaissance attempt by an intruder. It may also represent an advanced user attempting to determine the mailing address of a friend or co-worker.

3104 Archaic Sendmail Attacks

SubID Values: 0

Misc Field Info: "wiz" or "debug"

Recommended Alarm Value: 2

This sendmail attack is archaic and should not work against current versions of sendmail.

3200 WWW phf attack

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 5

This serious attack is used to exploit the phf program released with the NSCA and Apache web servers.

3201 WWW General cgi-bin attack

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 5

This signature detects any cgi-bin script that attempts to retrieve the file /etc/passwd.

3250 TCP Hijacking

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 5

This signature analyzes both streams of data within a TCP connection to detect when TCP hijacking may have occurred. This current implementation of this signature does not detect all types of TCP hijacking and false positives may occur. Even when hijacking is discovered, little information is available to the operator other than the source and destination addresses and ports of the systems being affected.

4000 UDP packet

SubID Values: 0 - 65536 Destination UDP port

Misc Field Info: none

Recommended Alarm Value: 1

This is event is used for logging UDP traffic. The token ***LevelOfTrafficLogging*** is used to configure the level of logging.

Level 1: No UDP logging occurs.

Level 2: This message records that the source host has sent a UDP datagram to the destination host using the UDP ports specified. The SubID for this signature is the destination UDP port. If the destination host does not have a UDP service at the requested port, an ICMP unreachable message will immediately follow. The default configuration is to not generate this signature unless it is explicitly specified within the configuration file.

Level 3: Same as level 2.

4001 UDP port sweep

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 5

This could be a serious attack. This signature identifies an attacking host that has sent several UDP datagrams to a number of different destination ports on the target host. This method is used by attackers to determine the services available on the target host for potential exploitation.

4100 TFTP Passwd File

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 5

Packets Required: A large sampling of UDP packets

This signature detects an attempt to access the passwd file via TFTP

6001 Normal SATAN probe

SubID Values: 0

Misc Field Info; none

Recommended Alarm Value: 5

This signature is generated when an attacking host has run the tool "SATAN" in normal mode against a target host on the network. Other types of attack activity similar to the method may also cause this signature to be generated.

6002 Heavy SATAN probe

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 5

This signature is generated when an attacking host has run the tool "SATAN" in heavy mode against a target host on the network. Other types of attack activity similar to the method may also cause this signature to be generated.

6050 DNS HINFO Request

SubID Values: 0

Misc Field Info: DNS query name

Recommended Alarm Value: 2

These signature detects an attempt to access HINFO records from a DNS server. This is commonly used by intruders to determine the types of systems on a network.

6051 DNS Zone Transfer

SubID Values: 0

Misc Field Info: DNS query name

Recommended Alarm Value: 1

These signature detects legitimate DNS zone transfers.

6052 DNS Zone Transfer from High Port

SubID Values: 0

Misc Field Info: DNS query name

Recommended Alarm Value: 4

These signature detects a DNS zone transfer with the source port not equal to 53. This is a common network reconnaissance technique used by intruders.

6053 DNS Request for all Records

SubID Values: 0

Misc Field Info: DNS query name

Recommended Alarm Value: 2

These signature detects a DNS request for all records.

6100 RPC Port Registration

SubID Values: RPC program number

Misc Field Info: none

Recommended Alarm Value: 5

This attack attempts to register new RPC services on a target host.

6101 RPC Port Unregistration

SubID Values: RPC program number

Misc Field Info: none

Recommended Alarm Value: 5

This attack attempts to unregister RPC services on a target host.

6102 RPC Dump

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 4

This is an example of network reconnaissance. This is produced by executing a "rpcinfo -p" against a target host.

6103 Proxied RPC Request

SubID Values: RPC program number

Misc Field Info: none

Recommended Alarm Value: 5

This attack exploits a vulnerability of the portmapper by causing it to forward RPC requests to the local system. This may defeat existing authentication mechanisms. The most widespread use of this vulnerability occurs in tricking mountd on NFS servers to disclose filehandles.

6150 ypserv Attempt

6151 ypbind Attempt

6152 yppasswdd Attempt

6153 ypupdated Attempt

6154 ypxfrd Attempt

6155 mountd Attempt

6175 rexd Attempt

SubID Values: 0

Misc Field Info: none

Recommended Alarm Value: 5

These signatures detect attempts to access the services specified. It is recommended that these services should not be accessed via the Internet.

Regular Expression-Based Signatures

Regular expression- (regex) based signatures are generated from stateful multi-layer datagram contents analysis. In simple terms, this means arranging the contents of packets and doing string matching on the results. Any regular expression up to 64 characters in length is supported, which provides an unlimited number of signatures.

While all regex based signatures have a **SigID of 8000,** each string must have a unique SubID assigned to it. Numbering SubIDs is left to the individual installing and maintaining NetRanger. Each string entry also specifies the port number and direction of traffic where it should be searched for. For example, in order to detect attackers using the command "help" on mailservers, the regular expression "[Hh][Ee][Ll][Pp]" should be reported only when it is in the datastream going to port 25 on a target host. This drastically lowers the number of false positives.

Performance is the only limitation to the number of strings that can be simultaneously searched for. Analyzing strings directed at the destination port provides substantially better performance than the reverse. For example, malicious activity inside of telnet sessions can be determined by what an attacker is typing, and not from what appears on the screen. A typical telnet session has over 95% of the network traffic generated going from the port 23 to the user's screen. In this example, better performance is achieved by only analyzing the remaining 5%.

NetSentry Based Signatures

NetSentry based signatures are generated by reporting packet failures and/or successes from NetSentry filters. When the NSG BorderGuard is used to enforce a specific security policy, an option exists to copy a subset of failed packets from the BorderGuard to the NSX. Each copied packet includes the name of the NetSentry filter that failed the packet. NetRanger matches this name with a pre-defined list of filter names and generates a security violation event record upon a match.

```
filter incom1_ip_spoof_fail
   ip_sa mask 0xFF000000 in (10.*.*.*)
      copy_to 10.1.5.3 35399
      fail;
end
```

The above filter was designed to analyze all packets entering a network. If the source address is from the network, it is an impossible packet and may be part of an IP spoofing attack. The filter will send a copy to the NetRanger/NSX at address 10.1.5.3, and then fail the packet. NetRanger will match the string "incom1_ip_spoof_fail" and generate a security violation event record. The event record will include the source and destination IP addresses and applicable ports.

Using this NetRanger capability, any filter on the BorderGuard is capable of generating event signatures. Because the BorderGuard allows a large number of highly configurable filters using the NetSentry filter language, the number of potential NetSentry event signatures generated by NetRanger could be considered unlimited. NetSentry filternames for versions 2.0 thru 3.1 of the BorderGuard OS is case sensitive. Version 4 of the BorderGuard OS and the ERS report filternames in uppercase. If the original filtername is "finger_fail", then NetRanger must be configured to look for the filtername "FILTER_FAIL".

All NetSentry based event records have a SigID of 10000. Each defined filtername in NetRanger must have a unique SubID assigned to it. Numbering SubIDs is left to the individual installing and maintaining NetRanger. The example security filters that come with NetRanger implement the following:

incom1_ip_spoof_fail	IP spoofing attacks
incom1_ip_source_route_fail	IP source routing attacks
incom1_tcp_frag_header_fail	Fragmented TCP header attacks
incom1_tcp_small_frag_fail	Undersized TCP header attacks
incom1_tcp_fail	Generic failed TCP connections
incom1_udp_fail	Generic failed UDP packets
incom1_ftp_fail	Failed FTP attempt
incom1_telnet_fail	Failed telnet attempt
incom1_smtp_fail	Failed e-mail attempt
incom1_dns_fail	Failed Domain Name Server attempt
incom1_gopher_fail	Failed gopher attempt
incom1_finger_fail	Failed finger attempt
incom1_www_fail	Failed WWW attempt
incom1_pop2_fail	Failed POP2 mail attempt
incom1_pop3_fail	Failed POP3 mail attempt
incom1_rpc_fail	Failed RPC attempt
incom1_auth_fail	Failed identd attempt
incom1_nntp_fail	Failed network news attempt
incom1_ntp_fail	Failed network time attempt
incom1_exec_fail	Failed rexec attempt
incom1_login_fail	Failed rlogin attempt
incom1_cmd_fail	Failed rsh attempt
incom1_printer_fail	Failed printer attempt
incom1_ntalk_fail	Failed network talk attempt
incom1_uucp_fail	Failed UUCP attempt
incom1_x11_fail	Failed X11 attempt
incom1_udp_dns_fail	Failed DNS packet
incom1_tftp_fail	Failed TFTP packet
incom1_udp_rpc_fail	Failed RPC packet
incom1_snmp_fail	Failed snmp packet
incom1_syslog_fail	Failed syslog packet

Default NetRanger Alarm Settings

This section lists the default NetRanger alarm settings. These settings provide for near-maximum Intrusion Detection but are not configured to provide automatic response. For each of the ID signatures, String Matches, Transport Events, and Policy violations, there is a configuration line in the file sensord.conf which allows you to set the automatic response and the reporting level. The default levels will give you some idea of the usual level of threat for each of these items. Certain sites will want to emphasize certain events by raising them to a higher level or turning on an automatic response, or to de-emphasize other events by lowering their reporting level.

Note: The SYN flood detector is configured to work on services that are set up for string match examination. If this is not appropriate for your installation, it may be changed in the file first.fil on the BorderGuard. Keep in mind that misconfiguring these settings may greatly increase the load on the sensor and it's portion of the network bandwidth without providing an increase in protection. Please consult a WheelGroup engineer for advice on this adjustment.

By default there is no auto response configured. To turn on auto-logging or auto-shunning for any of the following events, or to change their reporting level, you can edit the sensord.conf file or change the settings through the nrConfigure GUI.

The following Intrusion Detection signatures generate level 1 alarms by default:

IP options - Bad option list
IP options - Timestamp
IP options - Provide s,c,h,tcc
IP options - SATNET id
ICMP Echo Reply
ICMP Unreachable
ICMP Redirect (change a route)
ICMP Echo Request
ICMP Parameter Problem on Datagram
ICMP Timestamp Reply
ICMP Information Reply (obsolete)
ICMP Address Mask Reply
TCP connection records
UDP packet records
DNS Zone Transfer Request

The following Intrusion Detection signatures generate level 2 alarms by default:

IP options - Record packet route
ICMP Source Quench
ICMP Time Exceeded for a Datagram
ICMP Timestamp Request
ICMP Information Request (obsolete)
ICMP Address Mask Request

The following Intrusion Detection signatures generate level 3 alarms by default:

sendmail reconnaissance
Archaic sendmail attacks
DNS HINFO Request
DNS request for all records
ypserv attempt
ypbind attempt
yppasswdd attempt
ypupdated attempt
ypxfrd attempt

The following Intrusion Detection signatures generate level 4 alarms by default:

Fragmented ICMP traffic
Large ICMP traffic
smail attack
sendmail invalid recipient
sendmail invalid sender
DNS Zone Transfer from other port
RPC dump (rpcinfo -p)
rexd attempt

The following Intrusion Detection signatures generate level 5 alarms by default:

IP options - Loose source route
IP options - Strict source route
ICMP network sweep w/Echo
ICMP network sweep w/Timestamp
ICMP network sweep w/Address Mask
TCP port sweep
Half-open SYN attack
WWW phf attack
WWW general cgi-bin attack
TCP Hijacking
UDP port scan
Tftp passwd file attempt
Normal SATAN probe
Heavy SATAN probe
RPC port registration
RPC port unregistration
Proxied RPC request

The following Intrusion Detection string matches generate level 2 alarms by default:

"+ +" in a telnet Session
"+ +" in a rlogin Session
"Security" in a WWW Session
The following Intrusion Detection string matches generate level 4 alarms by default:
"RETR passwd" in an FTP Session
"/etc/shadow" in a telnet Session
The following Intrusion Detection string matches generate level 5 alarms by default:
"IFS=/" in a telnet Session
The following TCP events generate level 1 alarms by default:
Connection request- tcpmux
Connection request- echo
Connection request- discard
Connection request- systat
Connection request- daytime
Connection request- netstat
Connection request- chargen
Connection request- ftp-data
Connection request- ftp
Connection request- telnet
Connection request- smtp
Connection request- time
Connection request- whois
Connection request- dns
Connection request- gopher
Connection request- finger
Connection request- www
Connection request- link
Connection request- kerberos-v5
Connection request- supdup
Connection request- hostnames
Connection request- iso-tsap
Connection request- x400
Connection request- x400-snd
Connection request- csnet-ns
Connection request- pop-2
Connection request- pop3
Connection request- sunrpc
Connection request- uucppath
Connection request- nntp
Connection request- ntp
Connection request- netbios
Connection request- netbios
Connection request- netbios
Connection request- imap2
Connection request- NeWS
Connection request- xdmcp
Connection request- nextstep
Connection request- bgp
Connection request- irc
Connection request- imap3
Connection request- ulistserv

Connection request- exec
Connection request- login
Connection request- shell
Connection request- printer
Connection request- courier
Connection request- uucp
Connection request- pcserver
Connection request- kerberos-v4

The following UDP events generate level 1 alarms by default:

UDP traffic - echo
UDP traffic - discard
UDP traffic - daytime
UDP traffic - chargen
UDP traffic - time
UDP traffic - dns
UDP traffic - tftp
UDP traffic - gopher
UDP traffic - www
UDP traffic - kerberos-v5
UDP traffic - sunrpc
UDP traffic - ntp
UDP traffic - xdmcp
UDP traffic - bgp
UDP traffic - imap3
UDP traffic - ulistserv
UDP traffic - biff
UDP traffic - who
UDP traffic - syslog
UDP traffic - printer
UDP traffic - talk
UDP traffic - ntalk
UDP traffic - route
UDP traffic - nfs

The sample policy filters incom1.fil and incom2.fil are designed to identify incoming traffic and pass or fail it based on set conditions. For example if you specified a FTP server and a SMTP server during the NetRanger configuration the sample filters would be modified to allow these hosts as exceptions to the rules which fail incoming FTP and SMTP. you could also provide a list of allowed SMTP servers as easily as one. This sample policy is excellent for use if the NetRanger is protecting the access point between a LAN and an untrusted WAN. If the NetRanger is instead providing separation between the Finance and Engineering departments of your site, the policy would probably be much more loose.

The following policy violations from the sample policy filters incom1.fil and incom2.fil generate level 3 alarms by default:

General TCP failure
General UDP failure
ftp
telnet
smtp
DNS
gopher
finger
www
pop2
pop3
rpc
auth
nntp
ntp
exec
login
cmd
printer
ntalk
uucp
x11
udp dns
tftp
udp rpc
snmp
syslog
ntp
rip
icmp
tcp block
dns reply
last_block

The following policy violations from the sample policy filters incom1.fil and incom2.fil generate level 4 alarms by default:

large icmp fail

The following policy violations from the sample policy filters incom1.fil and incom2.fil generate level 5 alarms by default:

IP spoofing
IP source routing
TCP frag header attack
TCP small frag attack
shun

Data Privacy Facility

Included as part of the NetRanger package, the NSG BorderGuard, comes with powerful encryption capability that can establish a Virtual Private Network (VPN) between geographically remote sites over public networks.

At a minimum, WheelGroup recommends that you use encrypted sleeves to encrypt all traffic between remote networks using the NetRanger NSX Sensor and the primary network that should be running both the NetRanger NSX Sensor and the Director. You do not necessarily have to use encrypted links between any internal NSXs you may be using for which traffic is only crossing your internal private networks.

A good method to determine a necessity for encrypted sleeves is to trace the path of alarms and traffic traveling between two NSX systems and consider the security of the networks that cross. If you feel comfortable with those networks, it is probably not necessary to use encryption and incur the performance reduction on traffic traveling within the sleeve.

The NetRanger sensor is shipped with a default security posture which may need to be modified to suit the needs at your site. The security posture can be viewed as two enforcement mechanisms. The first is the intrusion detection mechanism. This will probably require little or no modification as it is appropriate for almost all sites. The second is a set of filters which enforce the policy set by your site. This may require extensive study and modification as the sample policy provided with the NetRanger system is pretty restrictive.

V The Security Analysis Package

Collection, Management, and Analysis of NetRanger Data

Conceptual Overview

The NetRanger Security Analysis Package (SAP) collects, manages, and analyzes data.

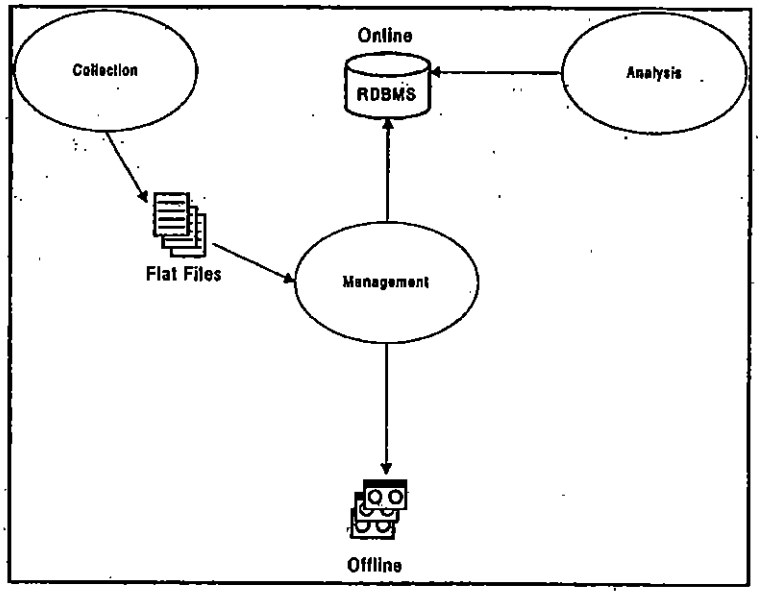


Figure V-1 : Overview of SAP Functions

Collection

SAP collects data into two basic types of log files: flat ASCII event files, and binary IP session logs. Data is written to files instead of databases for two fundamental reasons: speed and fault tolerance. Data can always be written to a flat file faster than a database. Flat files are always accessible—if the database is down, you've got nowhere to go.

- **Event logs** capture NSX alarms, system commands, and system errors. These are ASCII files written to the /usr/nr/var directory with the naming convention of log.<YYYYMMDDHHMM>. Event logs typically reside on a Director machine.
- **IP Session logs** capture all of the incoming and outgoing TCP packets associated with a specific connection. By definition, these logs contain binary data. They are written to /usr/nr/var/iplog and have the naming convention of iplog.<src IP address>. Session logs are usually left on a NSX system because the volume of session data is too great to transmit back to a Director without disrupting other NetRanger communications.

Management

There are three goals for data management, as follows:

- collecting data in a fast, fault-tolerant manner;
- staging that data onto database management and archival systems; and
- preventing disk systems from filling up through the use of proprietary file management techniques.

Analysis

Once data is staged, it can be analyzed for patterns and trends. Reports on such topics as network activity and vulnerabilities can also be generated. Although any number of different third-party tools can be used to generate these types of output, SAP is shipped with a small but comprehensive collection of Oracle SQL*Plus queries that show how event data can be analyzed relative to three basic dimensions: **time, space,** and **event**. These queries are described at the end of the chapter.

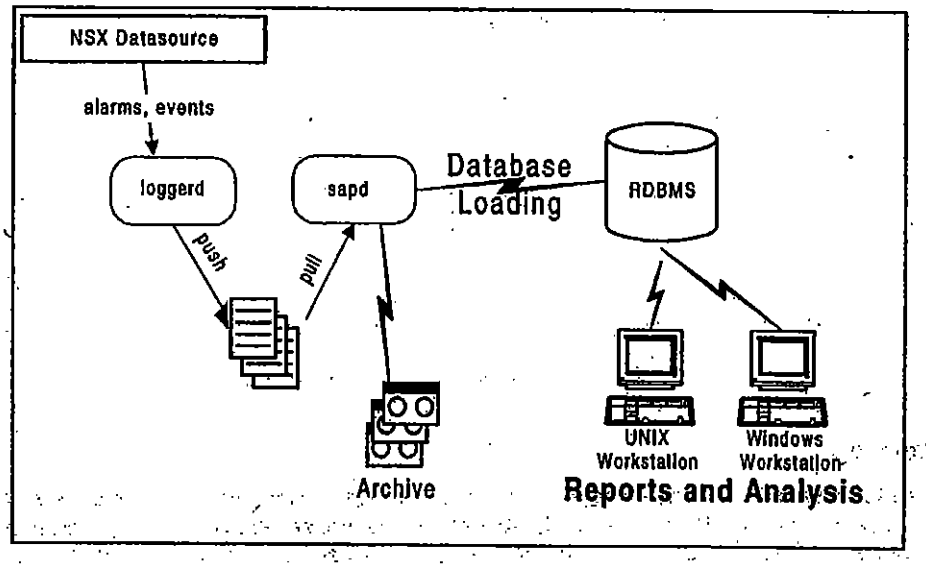


Figure V-2:Data Collection, Management, and Analysis

Function, Configuration, and Usage

This section focuses on how each SAP component works, how they can be configured, and how to use them.

NetRanger uses a simple push-pull process to migrate data from flat files onto a database. In the first part of that process, *loggerd* writes event, command, and error notifications into a single flat file in /usr/nr/var, which is serialized based on a configurable size, time, or combined threshold. The serialized file is moved to /usr/nr/var/new. In the second part of the push-pull process, *sapd* pulls the oldest file into /usr/nr/var/tmp and executes user-defined database load procedures. After the file is successfully loaded into a database management system, the log file is moved to /usr/nr/var/old, which is where old files are held until they can be dumped to a user-configurable destination (by default, an offline tape storage). *loggerd* also writes binary IP session logs to /usr/nr/var/iplog, which gets archived directly to dump.

sapd not only stages data, it also provides automated file management procedures that prevents an NSX or Director file system from filling up—otherwise, manual administration of the file system is required, which represents a type of system vulnerability.

The following figure illustrates the push-pull process.

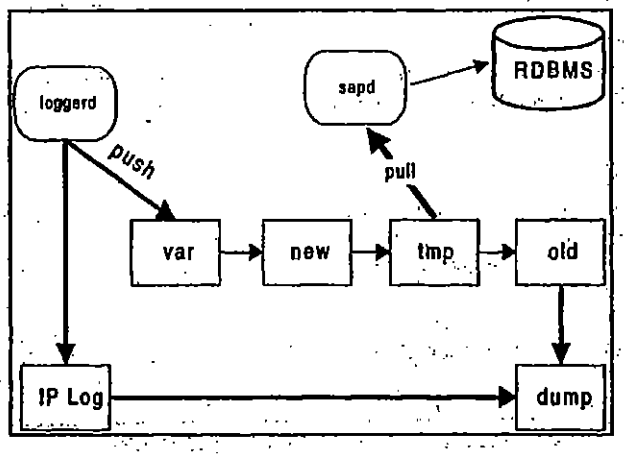


Figure V-3: The Push-Pull Process

Collection

loggerd

Data collection is controlled by *loggerd*, a daemon that serializes data by a time, size, or a combined time/size threshold. As mentioned previously, serialized event files are pushed from /usr/nr/var into /usr/nr/var/new. *loggerd* also writes binary IP Session logs to /usr/nr/var/iplog. *loggerd* also employs failsafe procedures to prevent runaway serialization and file overwriting.

The time threshold is specified via *loggerd*'s **NumberOfSwitchMinutes** token, and the size threshold is specified by the **NumberOfSwitchBytes** token. Values for these thresholds depend on such factors as NSX positioning on the network, network activity, and local file system free space.

In the following example, all but one file was serialized on the basis of a time threshold of one hour. The file log.199611070819 was serialized because the 100 kilobyte threshold was exceeded before the hour threshold was reached.

```
FILESIZE  DATE   TIME  FILENAME
   61058 Nov  7 06:19 log.199611070519
   60524 Nov  7 07:19 log.199611070619
   70008 Nov  7 08:19 log.199611070719
  103730 Nov  7 08:40 log.199611070819
   76926 Nov  7 09:40 log.199611070840
   83476 Nov  7 10:40 log.199611070940
```

Fail-Safe Features

One of the primary requirements of *loggerd* is that have fail-safe procedures under heavy loads. These fail-safe measures guard against runaway serialization and loss of files due to overwriting.

Runaway Serialization

loggerd automatically forces a minimum switch time of one minute. For example, even if the size threshold is triggered, *loggerd* will not serialize a log file until it has been open for at least 60 seconds. This feature eliminates runaway serialization during heavy network traffic.

In the following example, very heavy traffic occurred between 10:14 and 10:16. The log file that was opened at 10:14 was forced to grow to 259K because the one-minute threshold had not been satisfied.

```
FILESIZE  DATE   TIME  FILENAME
  100358 Nov 11 09:49 log.199611110916
  101432 Nov 11 10:14 log.199611110949
  259555 Nov 11 10:15 log.199611111014
  177152 Nov 11 10:16 log.199611111015
  100307 Nov 11 10:44 log.199611111016
```


Loss of Files from Overwriting

loggerd also prevents files from being overwritten when NetRanger daemons are started and stopped more than once a minute. *loggerd* simply moves log files with similar names into the /usr/nr/var/new area and gives them a unique file extension.

In the following example, nrstart and nrstop were executed in rapid succession and the NetRanger daemons started and stopped six times in a minute (16:44). As a result, *loggerd* appended the numbers 1, 2, and 3 to the end of each file:

```
FILESIZE  DATE   TIME  FILENAME
     178 Nov  8 16:44 log.199611081644
     313 Nov  8 16:44 log.199611081644.1 ←
       0 Nov  8 16:44 log.199611081644.2 ←
     306 Nov  8 16:44 log.199611081644.3 ←
```

Token Execution

Use the SwitchFile token to execute a switch/serialization on demand, bypassing configured thresholds. Other *loggerd* tokens should not be executed.

Management

sapd

sapd is the NetRanger daemon that launches user-defined actions for data and file management.

As stated in the beginning of this section, *sapd* pulls serialized data files from /usr/nr/var/new and loads them into a database management system. Time and size thresholds are evaluated sequentially until one of them is exceeded by the log file currently being written to in /usr/nr/var. This is controlled by *sapd* FileMgmt tokens. Every threshold can be thought of as a trigger, and every trigger has one of the following actions:

- DBLOAD, which loads the contents of an ASCII file into a database management system, such as Oracle or Remedy ARS.
- ARCHIVE, which loads ASCII or binary data into a user-defined output type, such as tape storage.

These actions are scheduled by a Launch Control process, which spawns these processes in the order that they are triggered. The following figure illustrates the trigger evaluation process.

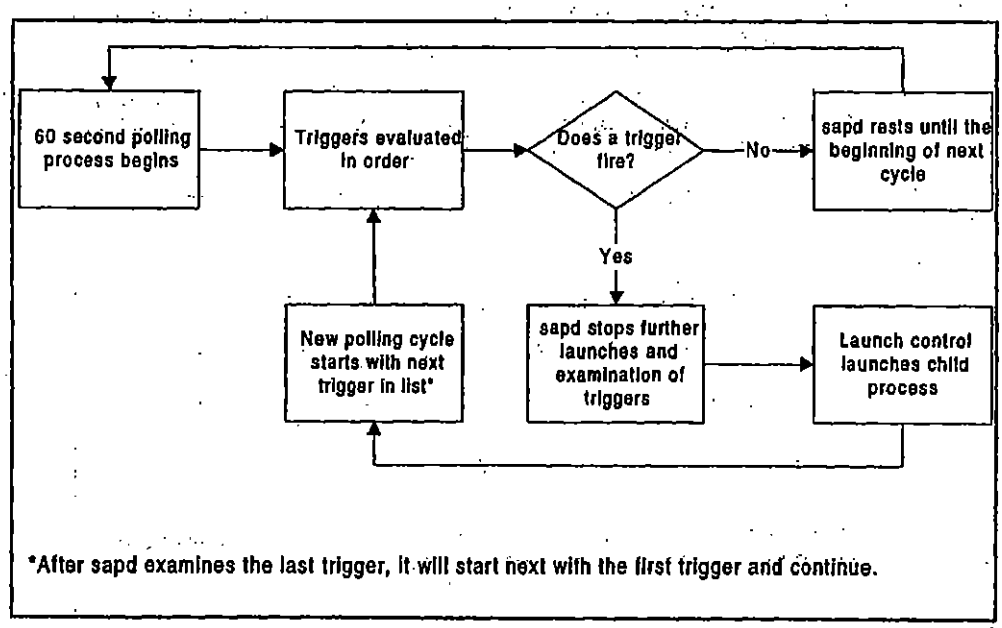


Figure V-4:Trigger Evaluation Process

Triggers
A trigger is composed of type, value, and action elements. The following table defines these elements.

FileMgmt Tokens	Type	Value	Action
DIRFILES	maximum number of log files allowed in a directory	non-zero positive integer (# of files)	DBLOAD ARCHIVE
DIRSIZE	maximum number of bytes that all of the log files in a directory can consume	non-zero positive integer (in units of Bytes)	DBLOAD ARCHIVE
DIRUSED	maximum percentage of disk partition that can be used by log files	integer between 1 and 99	DBLOAD ARCHIVE
IDLETime	*sapd* polling interval	non-zero positive integer (minutes)	N/A

For each FileMgmt token defined, *sapd* compares its value with the current state of the directory location. Its associated action is launched if its trigger value is exceeded. Although the only actions currently supported by *sapd* are DBLOAD and ARCHIVE, the underlying processes are fully configurable by the user.

Launched Actions
Both the DBLOAD and ARCHIVE processes are based on incremental steps that require a prior step to succeed before the next step is invoked. If a given step fails, the work associated with that step is rolled back and an error is logged. The steps in both of these processes are summarized on the following page.

DBLOAD Process

The DBLOAD process encompasses three basic states, PRE, RUN, and POST. Error conditions are handled by a fourth process, UNDO. The actual state transitions are defined as follows:

1. *sapd* starts the processing cycle with PRE.
2. PRE moves the file from /usr/nr/var/new to /usr/nr/var/tmp.
   - If PRE is successful, RUN is launched.
   - If PRE is not successful, nothing more is done.
3. RUN takes the oldest log file in /usr/nr/var/tmp, divides it into three separate subfiles based on record type (alarm event, system command, system error) and launches the user-defined DBLOAD program (by default, *sapx*).
   - If RUN is successful, POST is launched.
   - If RUN is not successful, UNDO is launched.
4. POST cleans the files in /usr/nr/var/tmp and moves the log file that was processed into /usr/nr/var/old.
5. UNDO restores to the state before PRE was launched and moves the original log file from /usr/nr/var/tmp back into the /usr/nr/var/new area.

The Control Flow can be summarized as follows:

PRE -> RUN -> POST (no errors, log file is loaded and moved to old)

or

PRE -> RUN -> UNDO (error occurred in RUN, revert to log file state before PRE)

DBLOAD loads the event log files into your local database. The various components that control execution of DBLOAD sub-system are as follows:

Tokens	ControlPre, ControlRun, ControlPost, ControlRunUndo
Programs	*sapx*
Scripts	load_pre.sh, load_run.sh, load_post.sh, load_undo.sh
Triggers	launch DBLOAD process when /usr/nr/var/new contains at least one file (that was serialized by *loggerd*)

ARCHIVE

ARCHIVE loads ASCII or binary data files into a user-defined output type, such as tape storage. The ctl_dump.sh script launched by the ARCHIVE trigger can be summarized as follows:

- /usr/nr/var/old reaches a certain byte count
- /usr/nr/var/iplog reaches a certain byte count
- launch NOTIFY process when /usr/nr/var/dump reaches a certain threshold

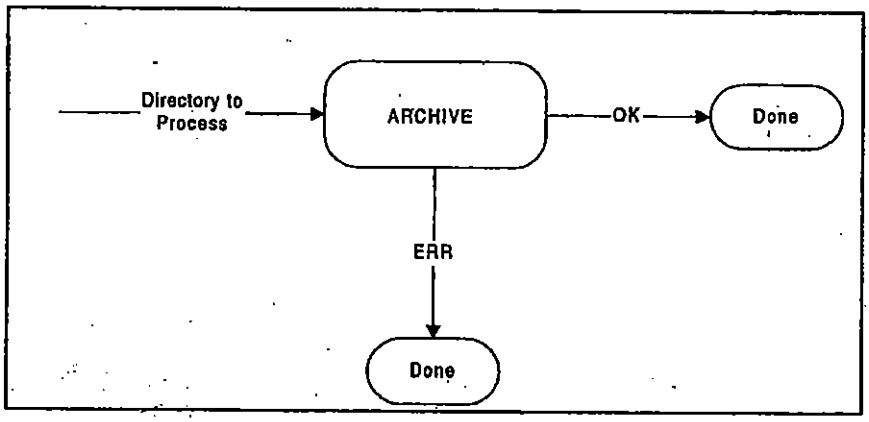


Figure V-5: Archive Process

Token Execution

Use VarSize to summarize `ls` information about vital directories. Use RunHist to show the launch history of DBLOAD and ARCHIVE. To execute these, use nrget.

sapx

sapx is the default data loader program shipped with the SAP package. It currently supports Oracle and the Remedy ARS system. *sapx* can load logs (error, command, alarm) and configuration files (signatures) into Oracle. It can also load logs (alarm) into Remedy ARS.

For *sapx* to run properly, the following parameters must exist: an input file (controlled by the LogFileName token), input and output types (SourceType and TargetType tokens respectively), and *sapx* messages.

The **LogFileName** token determines the path and filename to the log file to be processed. The **SourceType** token is a number between two and five that designates the log file type (2=errors, 3=commands, 4=events/alarms, 5=signatures).

Finally, the **TargetType** token is either a one or three, and designates what type of output to load into (1=Oracle, 3=Remedy). To configure this token, use the /usr/nr/bin/sap/skel/populate_sigs.sh. Type 5 loads of the /usr/nr/etc/signatures need to be performed only on initialization and when that file changes.

Refer to the following figure for a sample *sapx* message.

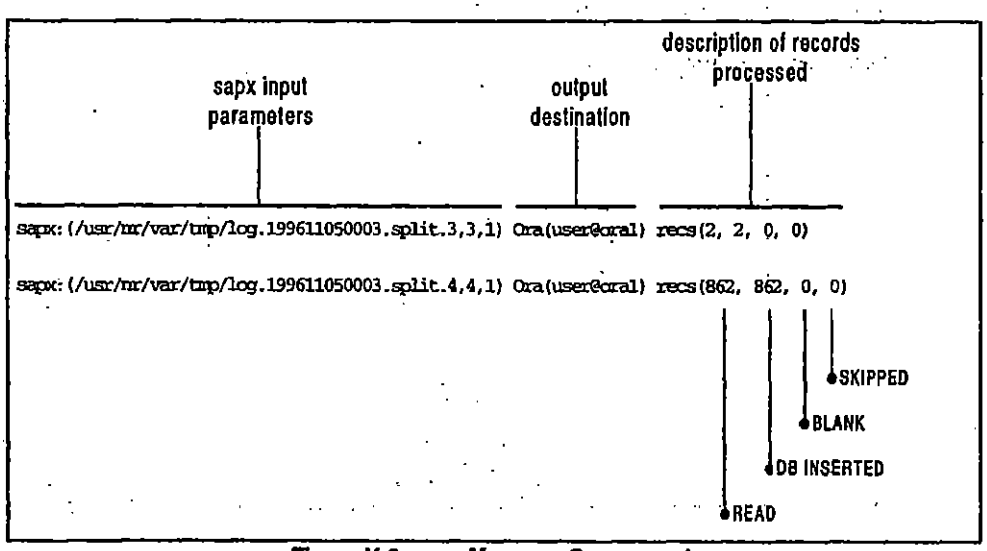


Figure V-6: sapx Message Components

Scripts and skel

The DBLOAD process relies on two types of scripts: control and conversion. Control scripts move data through the SAP system. Conversion scripts transform data into new output formats, such as cpio tape formats, printer hard copy, relational database tables, etc.

skel is a directory under /usr/nr that contains templates that specify alternate loading procedures for database types other than Oracle and Remedy ARS. These templates are effectively schema definitions for alternate bulk loaders.

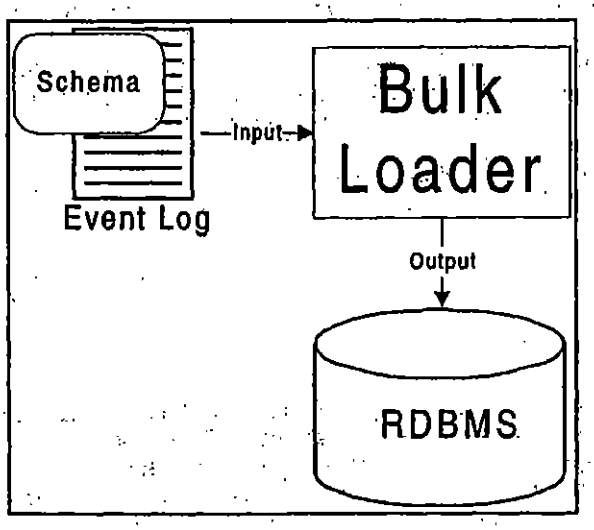


Figure V-7: Bulk Loading Process

Alternate Loading Procedures

Defining what gets Loaded

sapx, by default, attempts to load error, command, and alarm record types. If you do not want one of these record types to be exported to your target database, modify the "for filetype in" command in the script /usr/nr/bin/sap/load_run.sh.

For example,

```
for filetype in 2, 3, 4
```

can be modified to receive only alarms by deleting the entry types for "2" (commands) and "3" (errors):

```
for filetype in 4
```

Changing the Target Database from Oracle to Remedy ARS

Replace the control script in /usr/nr/bin/sap/load_run.sh with the control script in /usr/nr/bin/sap/skel/load_run.sh.remedy.

Alternate Database Schemas and Destinations
Use the event log templates in /usr/nr/skel/sap as a guide for changing the schema of the database load process, or as a guide for exporting event log data to databases other than Oracle and Remedy ARS. Refer to the following for further information:

- Use the create_log*.SQL files as a template for the schema declaration.
- Use the oraldr_log*.ctl files as a template for SQL*Loader field mappings.
- Use the load_run.sh.oraldr as an example load_run.sh control script.

One of the requirements for the DBLOAD scripts is that they must return an exit status. Otherwise, *sapd* will not be able to determine when an error has occurred or perform a UNDO. Use the following settings for returning an exit status:

- If the script completed without error, return 0
- If an error was encountered during processing, return 1 or greater

Analysis

sapr
sapr is a collection of SQL queries that generate simple columnar reports based on three different ways of viewing security-related data: Space, Time, and Event. In traditional database parlance, these are known as dimensions, where each dimension contains one or more elements. For example, the NetRanger Time dimension contains two subdimensions (Greenwich Mean Time and Local Time), which in turn contain the elements year, month, and so on. A breakdown of these dimensions, subdimensions, and elements follow.

Space
This dimension has the following two subdimensions: Location and Direction. The elements for each of these subdimensions are as follows.

- LOCATION. This identifies an event's source/destination addresses and ports
  - source address (SA)
  - source port (SP)
  - destination address (DA)
  - destination port (DP).
- DIRECTION. This refers to the flow of traffic relative to internal (IN) and external (OUT) IP addresses:
  - IN-IN
  - IN-OUT
  - OUT-IN
  - OUT-OUT

Time
This dimension has the following two subdimensions: LOCAL and GMT timestamps. The LOCAL timestamp defines occurrences relative to a locale, whereas GMT identifies events within a global context. Tracking both LOCAL and GMT timestamps simplifies data analysis once data has been loaded into a database. Queries can then be used to analyze local as well as global patterns. Both of these timestamps contains the following elements: year, month, day, hour, and minute.

Event
This dimension has four subdimensions: SIG NAME, SIG LEVEL, SIG STRING MATCH, and SIG ALARM STRING.

- SIG NAME defines a alarm signature's name and reference number.
- SIG LEVEL defines a alarm signature's severity level, which by default ranges from 1 to 5. Please refer to Chapter 3 for a complete overview of these signature levels.
- SIG STRING MATCH identifies the signature string that is searched for (e.g., "confidential").
- SIG ALARM STRING is the string that compliments the alarm (but only on certain signatures).

All event records have a SIG LEVEL and SIG NAME. Only certain signatures trigger the string data to be associated with the record.

Dimension Combinations
The Time, Space, and Event dimensions can be combined. They combine left to right, ending with a conditional selection criteria (trailer). The following tables summarize the combinations and how the three dimensions overlap.

Space Summary

space traffic, generic	(Space) + (trailer)
space traffic, event only	(Space) + (Event) + (trailer)
space traffic, event & time	(Space) + (Event) + (Time) + (trailer)
space traffic, time only	(Space) + (Time) + (trailer)
space traffic, time & event	(Space) + (Time) + (Event) + (trailer)

Event Summary

event traffic, generic	(Event) + (trailer)
event traffic, space only	(Event) + (Space) + (trailer)
event traffic, space & time	(Event) + (Space) + (Time) + (trailer)
event traffic, time only	(Event) + (Time) + (trailer)
event traffic, time & space	(Event) + (Time) + (Space) + (trailer)

Time Summary

time traffic, generic	(Time) + (trailer)
time traffic, space only	(Time) + (Space) + (trailer)
time traffic, space & event	(Time) + (Space) + (Event) + (trailer)
time traffic, event only	(Time) + (Event) + (trailer)
time traffic, event & space	(Time) + (Event) + (Space) + (trailer)

Table of Overlaps

Combination	Space	Event	Time
generic	x	x	x
event only	x		x
event &time	x		
time only	x	x	
time & event	x		
space only		x	x
space & time		x	
time & space		x	
space & event			x
event & space			x

SQL

Default Schemas

The native Oracle schema supported by *sapx* and sapr are illustrated in the following diagram. Each box on the left identifies how event data is group prior to being loaded into a database. Each box on the right represents the target database tables. The schemas for these tables are defined on the following pages.

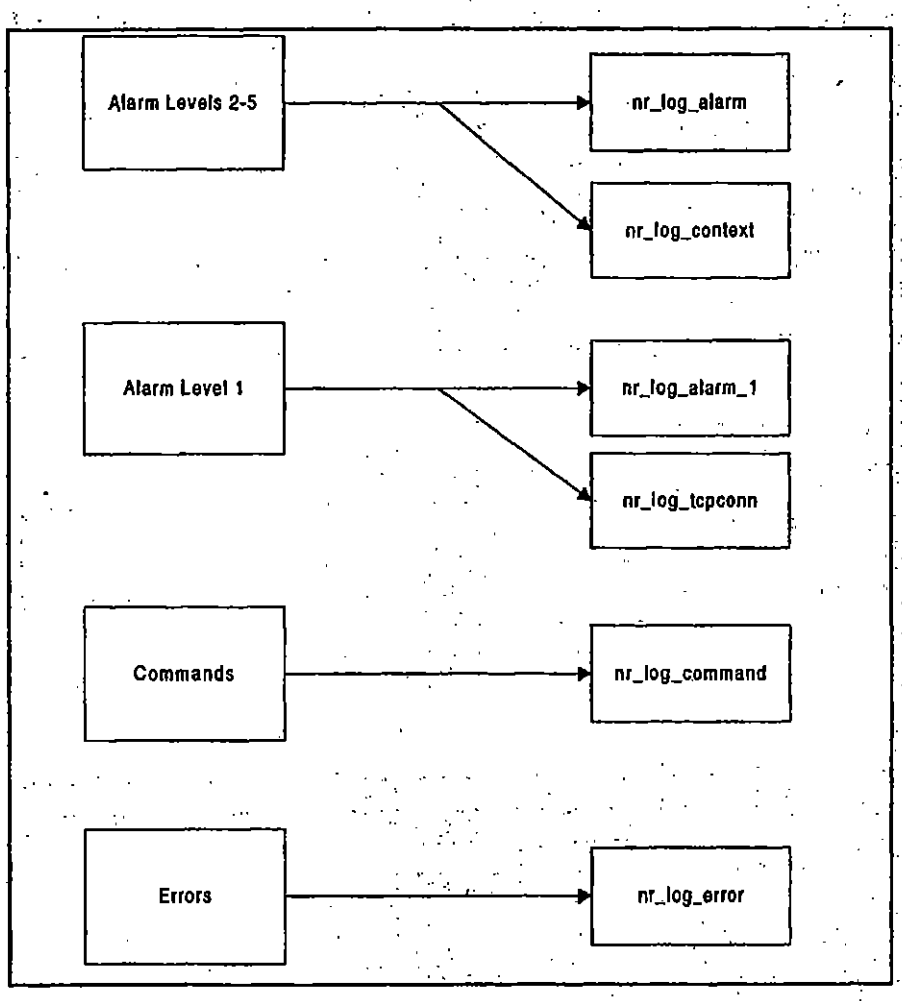


Figure V-8: Mapping of Event Data to Database Tables

nr_log_alarm and nr_log_alarm_1

Name	Type
EVENT_PROTOCOL	NUMBER(5)
RECORD_ID	NUMBER(10)
EVENT_DATE_GMT	DATE
EVENT_DATE-LOCAL	DATE
APP_ID	NUMBER(5)
HOST_ID	NUMBER(10)
ORG_ID	NUMBER(10)
FROM_STATE	CHAR(1)
TO_STATE	CHAR(1)
EVENT_LEVEL	NUMBER(5)
IP_SIGNATURE	NUMBER(10)
IP_SUB_SIGNATURE	NUMBER(10)
NETWORK_PROTOCOL	CHAR(3)
SRC_IP_ADDR	CHAR(32)
DST_IP_ADDR	CHAR(32)
SRC_PORT	NUMBER(5)
DST_PORT	NUMBER(5)
ROUTER_IP_ADDR	CHAR(32)
DATA_ALARM	CHAR(64)

nr_log_context

Name	Type
RECORD_ID	NUMBER(10)
EVENT_DATE_GMT	DATE
HOST_ID	NUMBER(10)
ORG_ID	NUMBER(10)
DATA_MATCH	CHAR(64)
DATA_INCOM	VARCHAR2(768)
DATA_OUTGO	VARCHAR2(768)

nr_log_tcpconn

Name	Type
RECORD_ID	NUMBER(10)
EVENT_DATE_GMT	DATE
HOST_ID	NUMBER(10)
ORG_ID	NUMBER(10)
IP_SUB_SIGNATURE	NUMBER(10)
SRC_IP_ADDR	CHAR(32)
DST_IP_ADDR	CHAR(32)
SRC_PORT	NUMBER(5)
DST_PORT	NUMBER(5)
ROUTER_IP_ADDR	CHAR(32)

nr_log_error

Name	Type
EVENT_PROTOCOL	NUMBER(5)
RECORD_ID	NUMBER(10)
EVENT_DATE_GMT	DATE
EVENT_DATE_LOCAL	DATE
APP_ID	NUMBER(5)
HOST_ID	NUMBER(10)
ORG_ID	NUMBER(10)
DATA_ERR	VARCHAR2(256)

nr_log_command

Name	Type
EVENT_PROTOCOL	NUMBER(5)
RECORD_ID	NUMBER(10)
EVENT_DATE_GMT	DATE
EVENT_DATE_LOCAL	DATE
APP_ID	NUMBER(5)
HOST_ID	NUMBER(10)
ORG_ID	NUMBER(10)
SRC_APP_ID	NUMBER(5)
SRC_HOST_ID	NUMBER(10)
SRC_ORG_ID	NUMBER(10)
DATA_CMD	VARCHAR2(256)

Default Queries

Default queries have the following file naming convention:

FILENAME = DIMENSION + NUMBER + SPECIAL

Where DIMENSION = event, space, and time; NUMBER = unique number of this dimension's report; and SPECIAL has two variables, either u (denotes user input required) and x (denotes subordinate query not to be called by user).

Queries that do not require user input (i.e., those without any special indicators) are non-blocking and can be called from a batch control process.

Event Dimension Queries

event1.SQL	SIGNAME + COUNT
event2.SQL	LEVEL + COUNT
event3.SQL	STRING + COUNT
event4.SQL	STRING + SIGNUM + COUNT
event5u.SQL	STRING + SIGNUM + COUNT (with USER criteria)

Space Dimension Queries

space1.SQL	SRC + DST + COUNT
space2.SQL	SRC + DST + SIGNUM + COUNT
space3.SQL	SRC + DST + STRING + COUNT
space4.SQL	SRC + DST + STRING + SIGNUM + COUNT
space5u.SQL	SRC + DST + SIGNUM + COUNT (with user criteria)

Time Dimension Queries

time1.SQL	driver for time1x zoomtime query
time1x.SQL	zoomtime query statement (count per time slice)
time2.SQL	driver for time2x zoomtime query
time2x.SQL	zoomtime query statement (count of sigs per time slice)
time3.SQL	driver for time3x zoomtime query
time3x.SQL	zoomtime query statement (count of src,dst per time slice)
time4.SQL	driver for time4x zoomtime query
time4x.SQL	zoomtime query statement (count of src,dst,sig per time slice)
time5.SQL	driver for time5x zoomtime query
time5x.SQL	zoomtime query statement (count of src,dst,string per time slice)
time6u.SQL	driver for time6x zoomtime query
time6x.SQL	zoomtime query statement (count of src,dst,string per time slice with user criteria)

SQL Syntax & Methods: Examples in the Queries

File	Shows how to...
event1.SQL	join with nr_sigs to get sig name instead of just sig number
event3.SQL	ignore blank strings
event5u.SQL	prompt user for ORDER BY clause
time6u.SQL	1) prompt user for WHERE SIG= clause. 2) pass parameters to subordinate routine
space5u.SQL	1) prompt user for WHERE IPADDRESS= clause. 2) string search with LIKE and %

Example Queries

```
-- event3.sql :: STRING + COUNT

select
substr(data_alarm,1,48),
count(*) "COUNT"
from nr_log_alarm
where length(rtrim(data_alarm)) > 1
group by data_alarm
order by 2 desc;
```

```
-- space2.sql :: SRC + DST + SIGNUM + COUNT

select
substr(src_ip_addr,1,18) "SOURCE",
substr(dst_ip_addr,1,18) "DEST",
ip_signature "SIGNUM",
count (*)
from nr_log_alarm
group by src_ip_addr, dst_ip_addr, ip_signature
;
```


FileMgmt Tokens

Tokens	Default
Tokens for Directory Paths[a]	
LogFilePathNew	/usr/nr/var/new
LogFilePathTmp	/usr/nr/var/tmp
LogFilePathOld	/usr/nr/var/old
LogFileMask	log.
LogFilePathDump	/usr/nr/var/dump
LogFilePathIPLog	/usr/nr/var/iplog
LogFilePathVar	/usr/nr/var
Tokens for DBLOAD Launch Scripts[b]	
ControlPre	/usr/nr/bin/sap/load_pre.sh
ControlRun	/usr/nr/bin/sap/load_run.sh
ControlPost	/usr/nr/bin/sap/load_post.sh
ControlRunUndo	/usr/nr/bin/sap/load_undo.sh
Oracle Tokens[c]	
DBUser	username for oracle access
DBPass	password for DBUser
Tokens for Remedy or other DBMS Default[d]	
DBUser2	username
DBPass2	password for DBUser2
DBAux1	auxiliary environment information
DBAux2	auxiliary environment information
DBAux3	auxiliary environment information

[a]Using these defaults is recommended, but they can be changed if necessary.

[b]These tokens can be changed, either permanently or on the fly to facilitate custom operations. See documentation for *configd* and Java configuration for information about token operation.

[c]These must be set before you can connect to Oracle. They are set to match whatever user/password was specified in the DBMS setup (create user).

[d]These tokens are included to provide extra flexibility for non-Oracle databases.

Appendix A Troubleshooting

Overview

Sometimes, things go wrong. Components often display confusing and unexpected error messages, or behave outside their normal parameters. Identification of these symptoms, their possible causes, and their possible solutions could potentially be a time-consuming and frustrating experience.

Ideally, the material in this appendix will provide a framework to help you identify and resolve routine problems. When problems are not so routine, this appendix will also help you by ruling out possibilities before you call WheelGroup Technical Support. This will ensure a faster and more beneficial resolution to your problem.

The Troubleshooting appendix consists of the following sections:

- Director Running
- Director Not Running
- Connectivity
- NSX
- Oracle

Director Not Running

Symptom	Possible Cause(s)	Possible Solution(s)
The following error message is displayed when trying to start the Director system from an HP Terminal session: `Cannot write message to Director, errno = 2`	This error occurs when *smid* tries to write to a socket that does not exist. This may occur either because: The Director's nrdirmap application has not created its communication socket in /usr/nr/tmp because the OpenView user interface (ovw) was not started.	First ensure that the underlying NetRanger services, such as *postofficed* and *smid* are running. The easiest way to accomplish this is to execute the following from the console command line: /usr/nr/bin/nrstatus Run /usr/nr/bin/nrstart as user netrangr if these services need to be started. Then bring up the OpenView user interface (ovw) by executing `$OV_BIN/ovw &`. This will automatically start nrdirmap.
The following error message is displayed when trying to start the Director system from an HP Terminal session: `Cannot write message to Director, errno = 233`	This error message is generated when *smid* writes to a socket whose buffer is overflowing. This can occur when the Director's nrdirmap application is not running.	Ensure that the OpenView user interface (ovw) is running by executing `$OV_BIN/ovw &`. This will automatically start nrdirmap.
The following error message is displayed when trying to start the Director system from an HP Terminal session: `Cannot write message to Director, errno = 239`	This error message occurs when *smid* and nrdirmap do not have adequate permissions to talk to one another via sockets in /usr/nr/tmp.	Ensure that the *smid* process is owned by user netrangr, and that nrdirmap runs as suid netrangr.
One of the following error messages is generated by HP OpenView upon startup: `848967818198: WgcSema can't access '/usr/nr/etc/nrdirmap.semaphore' Error: nrdirmap:main, semaphore initialization failed.`	The nrdirmap.semaphore file in /usr/nr/etc has been deleted or has improper permissions.	Ensure that /usr/nr/etc/nrdirmap.semaphore exists and that the netrangr *user* and group accounts have read access to the file. If the file does not exist, you can create the file with an ASCII editor, such as *vi*. Add a single character to the file and save it in /usr/nr/etc with the name nrdirmap.semaphore. Also ensure that /usr/nr/bin/nrdirmap runs as suid netrangr.

Director Not Running (continued)

Symptom	Possible Cause(s)	Possible Solution(s)

`nrdirmap: fatal: libovw.so.1: can't open file: errno=2`	The LD_LIBRARY_PATH environment variable is not set properly in your user environment. Note that this may indicate that you are logged onto the Director platform as the wrong user.	Follow the instructions in Chapter III for setting up an HP OpenView environment for user accounts other than netrangr.
		
		If the user account is based upon either the bourne or korn shell the following lines should exist in the user's $HOME/.profile:
		
		`if [ -d /opt/OV ] ; then`
		`. /opt/OV/bin/ov.envvars.sh`
		
		`PATH=$OV_BIN:$PATH`
		`export PATH`
		
		`LD_LIBRARY_PATH=$OV_LIB:$LD_LIBRARY_PATH`
		`export LD_LIBRARY_PATH`
		`fi`
		
		Please refer to Chapter III if the user account is based on the cshell.

Director Running

Symptom	Possible Cause(s)	Possible Solution(s)
The Director's security map contains an NSX icon, but fails to show any events for that NSX.	The Director's **Severity Status** attributes are set higher than the level of alarms being generated by the NSX.	From the Director interface, highlight the icon for the NSX system and then either press **Ctrl+O** or select **Edit->Describe->Modify** from the menu bar. Then select **NetRanger/Director** and press **View/Modify**. Ensure that the *Minimum Marginal* and *Minimum Critical* status thresholds are low enough to register events from the NSX in question.

	The level of alarms generated by the NSX system fall below the **routing threshold** set in the /usr/nr/etc/destinations file.	If the Director Severity Status thresholds are set properly, ensure that the routing threshold in the NSX's /usr/nr/etc/destinations file is set low enough to route information to the Director. In the following example, the destinations file has the Director's *smid* event level set to "4", which means that only alarm events of level 4 and 5 will be forwarded onto the Director platform running on director1.wheelgroup:
		
		`1 datastore.wheelgroup loggerd 1 EVENTS,ERRORS,COMMANDS`
		`2 director1.wheelgroup smid 4 EVENTS`
		
		The Director should begin displaying alarms properly if you reset firefly's alarm threshold from "4" to "2".

An NSX's alarms are properly displayed on the Director security map, but information on those alarms does not appear in the **Show Current Events** window and the event log file in the Director's /usr/nr/var directory does not contain any records from that NSX.	The Director *loggerd* daemon is not listed as a destination in either the NSX's or the Director's configuration files.	You can either create an entry in the NSX's /usr/nr/etc/destinations file for the Director's *loggerd* service, or you can create a **DupDestination** entry in the Director's /usr/nr/etc/smid.conf file to redirect event data to *loggerd* from *smid*. The latter would look as follows: `DupDestination <HostId>.<OrgId> loggerd 1 EVENTS,ERRORS,COMMANDS` Where <HostId>.<OrgId> are the Director's values (e.g., director1.wheelgroup). **Note:** The first option requires that the NSX's destination file contain two entries for the Director platform: one for *smid* and another for *loggerd*. This doubles the amount of network traffic that must be sent to the Director platform. Although the second option will create some overhead for *smid*, it is the preferred configuration, because it only requires one copy of event data to be sent over the communication pathway.

Director Running (continued)

Symptom	Possible Cause(s)	Possible Solution(s)
Information is properly displayed in the Director's **Show Current Events** window, but the mouse pointer turns into an hourglass and never changes back.	The current events utility continues to pull information from the Director log files as long as the window is up.	Press the **Stop** button to terminate the filtering application. You can then use the scrollbars and menu options to look at the data. Press the **Close** button when you wish to exit this window.
The following error message is displayed when you execute a Director menu function, such as **Security->Show Context**, or **Security->Show Names**: `/usr/bin/sh: /usr/nr/bin/<executable>: Execute permission denied.`	The user who brought up the HP OpenView user interface (ovw) does not have proper access permissions for the /usr/nr subdirectories.	The only users authorized to use NetRanger Director utilities are: • the user netrangr, and • users in the group netrangr If you want a user account other than netrangr to be able to run all the Director utilities, you must assign that user to the *group* netrangr. On HP systems, use the SAM utility; on Sun systems use admintool. Note: On HP Systems the user netrangr primary group may not be netrangr because the group ID listed in /etc/passwd does not match the netrangr group ID in the /etc/group file. This can be corrected by typing the following command: `newgrp - netrangr` Sun users do not have to do this. Refer to Chapter IV for more information on how to enable and disable nrdirmap.

Your Director system is receiving the following error messages over and over in it's /usr/nr/var/errors.postofficed file:
 	
`Net received spoofed msg, discarding message from <IP addr>, errno=0`
 	
`Net received message from unknown host <IP addr>`
`Discarding message from <AppId>.<HostId>.<OrgId>, errno=0`
	The /usr/nr/etc/routes file on your Director does not contain an entry for the HostId–OrgId identified in the error message.	Create the appropriate entries in the Director's *organizations*, *hosts*, and *routes* files that reside in /usr/nr/etc.

Connectivity

Symptom	Possible Cause(s)	Possible Solution(s)
Unable to access an NSX sensor or any of the NetRanger services running on the system.	Can you ping the NSX Sensor? Yes—This means that the sensor platform is powered up but that something is wrong with the NSX services.	Telnet onto the box as "netrangr" and run /usr/nr/bin/nrstop. Try to deduce why services were disrupted by examining whatever error files exist in /usr/nr/var. If errors are found refer to the appropriate Troubleshooting section, or contact Technical Support. Once the cause has been identified restart the NSX services by executing /usr/nr/bin/nrstart.
	No—The NSX system's BorderGuard is shunning the Director. This can occur when: a. an attack is run from the Director system (as part of a system test or demonstration) that causes the NSX to shun the Director system. b. the NSX does NOT contain a **NeverShunIPAddress** entry for the Director system.	Gain access to the NSX system to either issue an unshun command via local execution of nrset, or restart *managed*, which will clear all current shuns on the BorderGuard. In either case you need to gain access to the NSX sensor via one of the following means: a. connect to the NSX sensor via its modem b. connect to the NSX via its COM1 serial port c. connect to the NSX system's BorderGuard via the BorderGuard's console Although option "a" provides remote access the NSX system, options "b" and "c" require physical access to the NSX. Option "b" also requires the use of a notebook computer or a terminal device.

NSX

Symptom	Possible Cause(s)	Possible Solution(s)
Unable to start or stop the NetRanger daemon processes when running **nrstart** or **nrstop**.	You are trying to run these utilities from an account that does not have access rights to the NSX daemons.	Ensure you are logged onto the NSX or Director platforms under the same user account that was used to start its daemon services ("netrangr" by default).
/usr/nr/var/errors.smid contains multiple instances of the following error message: `socket received data from unknown host.`	The NSX system's *sensord* is receiving **copy_to** messages from a BorderGuard or Passport security device that has not been properly defined in /usr/nr/etc/sensord.conf.	Properly define a **RecordOfDataSource** for the BorderGuard or Passport device that is trying to send copy_to messages to the NSX system. Refer to Chapter III for how to properly define the **<IP address>** and **<IP mask>** values for this sensord.conf token.

Oracle

Symptom	Possible Cause(s)	Possible Solution(s)
Cannot determine if Oracle is installed.		Check local and mounted file systems using commands `df`, `mount`, `find`. Look for "oracle" and "product/7*".

Cannot determine if Oracle is running.		Use the following command:
		`ps -ef | grep ora`
		
		After entering the command, the following should appear:
		`oracle 360 1 0 Jun 14 ? 0:01 ora_pmon_NRDB`
		`oracle 362 1 0 Jun 14 ? 4:36 ora_dbwr_NRDB`
		`oracle 364 1 0 Jun 14 ? 3:13 ora_lgwr_NRDB`
		`oracle 366 1 0 Jun 14 ? 0:40 ora_smon_NRDB`
		
		This means that there is an Oracle instance of SID=NRDB running since June 14. The user account oracle is the owner of these processes.

Oracle Installer (orainst) was unable to find any products to install.	start.sh was not run prior to starting orainst.	Run start.sh to prepare your environment for the orainst program. Look for the following file: /cdrom/cdrom0/orainst/start.sh

`sqlplus: not found`
or
`sqlldr: not found`
	The Oracle bin directory is not present or specified properly in your `$PATH`.	Set `$PATH` to include $ORACLE_HOME/bin.

The following error message is displayed when you try to run sqlplus: `---/oracle/product/7.3.2/bin/sqlplus: cannot open`	The shell finds sqlplus, but it cannot be executed. This can occur when your `$PATH` includes references to the wrong versions of the Oracle binaries. For example, you have mounted the wrong Oracle directories from a file server. Therefore, you are trying to execute HPUX binaries on a SPARC platform.	Ensure that the $ORACLE_HOME directory contains the proper binaries for the platform you are running on. Refer to *Remote Setup: heterogeneous/homogeneous* in *Appendix B*.

sqlplus, sqlldr, or *sapx* fail with the following SID error message:
 	
`ERROR: ORA-01034: ORACLE not available`
`ORA-07200: slsid: oracle_sid not set.`
	The `ORACLE_SID` environment variable, which identifies which database instance to use, was not set properly prior to starting sqlplus.	Set `ORACLE_SID=NRBIG`.

Oracle (continued)

Symptom	Possible Cause(s)	Possible Solution(s)

sqlplus, sqlldr, or *sapx* fails with the following libc error message:
 	
`libc.so.xxx: can't do something`
	$ORACLE_HOME/lib is not part of the LD_LIBRARY_PATH environment variable.	Add ORACLE_HOME/lib to the LD_LIBRARY_PATH environment variable. If you are running either the bourne or korn shell ensure that your $HOME/.profile contains the following entries:
		
		`LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib`
		`export LD_LIBRARAY_PATH`

sqlplus, sqlldr, or *sapx* fail with the following TNS error message: `ERROR: ORA-12154: TNS: could not resolve service name`	Oracle cannot understand the name specified in your connect string.	Ensure that the Oracle file tnsnames.ora resides in its proper location (usually $ORACLE_HOME/admin/network) and that it is properly formatted. The use the tnsping utility to test sqlnet connectivity to your remote database.

sqlplus, sqlldr, or *sapx* return a TNS or USER/PASSWORD error message.	Improper connect string.	Correct the syntax on the connect string. Connecting from the shell prompt, do one of the following:
		
		If specifying the password on the command line, type:
		`sqlplus <user>/<password>@<host>`
		
		Otherwise, type:
		`sqlplus <user>@<host>`
		
		and sqlplus will prompt for a password.

Appendix B NetSentry and DPF Filter Settings

This section lists and explains the settings for all of the filter (fil), command (cmd), and data privacy facility (dpf) files required to deploy the BorderGuard or Passport as part of an NSX system. Please note that many of the settings described in this section have been automatically set for you by the nrconfig script shipped with NetRanger under the /usr/nr/bin directory. This information is presented here in the event you need to manually edit these files. Refer to the appropriate NSG BorderGuard Reference Manual for additional information about these files

Standard Files

The standard files that must be configured on a BorderGuard device are as follows:

- startup
- filters.cmd
- sleeves.dpf and sleeves.fil
- pubkeys.dpf
- shun.fil
- first.fil
- Incom(x).fil
- last.fil

startup—This is the boot configuration file that is read each time the BorderGuard powers up or resets. This file performs these tasks:

- names the device
- compiles and applying filters
- configures interfaces
- sets up route tables
- executes the files required to establish encrypted sleeves
- establishes a nameserver
- starts daemons such as *telnetd*, *gated*, and *snmpd*

filters.cmd—This file compiles all the filters that enforce your security policies. It is called in the startup file via the following command:

```
@filters.cmd
```

This file contains a single line for each filter used on the BorderGuard. Most filters.cmd files contain at least the following three entries:

```
NetSentry> compile_filter first.fil
NetSentry> compile_filter incom1.fil
NetSentry> compile_filter shun.fil
```

DPF Files

Please note that these filters only apply to the BorderGuard 1000 and 2000 systems. The Passport system currently does not provide any type of VPN Facilities.

Refer to the NSG manual, *Data Privacy Facility User's Guide*, before attempting to make changes to any of the files in this section.

sleeves.dpf—This file contains sleeve definitions for the BorderGuard's Data Privacy Facility (DPF). During initialization, it is loaded in as a set of sleeve definition commands and placed into the DPF interface's working storage. The default sleeve settings should rarely need editing. The only entries that should be edited are sleeve definitions. The recommended format for these entries is

```
s_<netid>_<netid>_<sleeve_num>
```

where `netid` corresponds to the network ID established in the /usr/nr/etc/hosts file discussed in *Appendix D The NSX File Structure* and `sleeve_num` corresponds to the sleeve number at a particular site. The IP addresses assigned for group1 and group2 should correspond to the router/bridge interface designated on the BorderGuard as the DPF Interface.

NOTE

Sleeve definitions should be identical at both the local and remote site. For example, consider site #1 with 2 NetRangers, whose routers have the DPF IP addresses 199.99.99.1 and 199.99.100.1, and site #2 with one NetRanger, whose router IP address is 198.98.98.1.

The following sleeve definitions might apply:

```
sleeve s_100_101_1 group1 199.99.99.1 group2 198.98.98.1
sleeve s_100_100_1 group1 199.99.99.1 group2 199.99.100.1
```


sleeves.fil—This file contains two filters for every DPF sleeve connection defined in *sleeeves.dpf*. The first filter controls outbound traffic and the second filter checks inbound traffic.

NOTE

The **IP address of the NSX** (after the *copy_to* statements) and the **sleeve name** must match the name designated in sleeves.dpf. For example, if site #1 has a class C address 199.99.99.0 and site #2 has a class C address 198.98.98.0, and the sleeve name is s_100_101_1, the filter would look something like the following:

```
filter dpf_100_101_1
    ip_sa in (199.99.99.*)
    set_sleeve s_100_101_1;
    end;
    filter dpf_fail_100_101_1
    ip_da in (199.99.99.*)
    not sleeve s_100_101_1
    icmp_unreachable
    copy_to IP_ADDRESS_NSX fail;
    end;
```

A corresponding filter would be created for the remote site and would replace 199.99.99.* with 198.98.98.* but the sleeve names would remain the same.

sleeves.cmd—This file compiles and applies all the filters required to establish encrypted sleeves between a local BorderGuard and one or more remote sites. The filters are established in the *sleeves.fil* file and should be applied so that all traffic traveling between two sites is encrypted. For example, if site #1 had a class C address 199.99.99.0 and site #2 had a class C address 198.98.98.0, then the apply statements would be similar to the following:

```
ip apply dpf_100_101_1 on 199.99.99.* to 198.98.98.*
ip apply dpf_fail_100_101_1 on 199.99.99.* to 198.98.98.*
```

In this example, `dpf_100_101_1` and `dpf_fail_100_101_1` correspond to the filter names established in the file *sleeves.fil*.

pubkeys.dpf—This file contains the public keys for the BorderGuard. This file should have an entry for the local system and each remote BorderGuard for which an encrypted sleeve is maintained.

NOTE

sleeves.dpf requires long keys. Be sure that you generate long public keys for both ends of the encrypted sleeve.

Filter Templates

These files enforce security policies as well as establish filtering on the BorderGuard/Passport. Refer to the *NetSentry User Guide* for information on how to create and apply filters. When you are editing these filters, you may find it helpful to refer back to the section in this chapter that you used when you defined your security policy.

shun.fil—NSX system uses this file to **dynamically block** outgoing as well as incoming IP addresses. It is not applied to a BorderGuard/Passport until the NSX generates an alarm for an attack signature that is configured to shun.

NOTE

The name of this file should NOT be changed. The only thing that needs to be edited in this file is the IP address of the NSX that follows the `copy_to` command. If you have run the NetRanger installation script *nsxinstall* this should already be changed.

first.fil—This file identifies which network traffic should be copied to the NSX sensor. First, change the IP address of the NSX that follows the `copy_to` command if that has not already been changed. Then change any string-matching requirements defined in sensord.conf (described further on in this section).

For string matching to be effective, all packets for a given service (i.e. TELNET at port 23) must be copied to port 35398. By default, FTP (21), TELNET (23), mail (25), DNS (53), and RPC (111) packets are all copied. To set up string matching on additional services such as WWW (80), or on FTP data (20), add the following command lines to first.fil:

```
# Copy certain TCP traffic
ip_protocol in (6)
tcp_dp in (21,23,25,53,111,80,20)
copy_to IP_ADDRESS_NSX 35398 break;
```

NOTE

The port number following the NSX is 35398 instead of 35399 as it appears in the other filter files. **This is the UDP port for intrusion detection. Do NOT change it!**

incom(X).fil—This file is the core of the NetRanger security system and it establishes the security policy defined earlier in this chapter. Apply this filter to each network interface to establish a security policy. The general naming convention is `incom1.fil` for interface 1, `incom2.fil` for interface 2, and so on.

NOTE

The default templates assign `incom1` to the interface connected to your untrusted networks. Notice that the number following the `copy_to` command in this file is 35399 instead of 35398, as indicated in the previous filter file. **This is the security policy monitoring port. Do NOT change it!**

If nrconfig has not been run, the first information that should be included is the IP address of the NSX after every `copy_to` command.

```
copy_to IP_ADDRESS_NSX 35399 break;
```

Next, make changes to the *incom(X)_ip_spoof_fail* filter. **This filter prevents and alarms any connections from outside your trusted network that have a source address from your internal network.** When applied to your trusted network interface, it assures that no outgoing traffic is spoofing an external address. Collect all of your internal class C and class B addresses and create an entry for each one. You should also create an entry for the individual external IP address. In the case of a site with two internal protected networks with class C addresses 199.99.99.0 and 199.99.100.0, and a router with an external IP address of 199.99.101.1, the filter attached to the untrusted network would look like the following:

```
filter incom1_ip_spoof_fail
ip_sa mask 0xFFFFFF00 in (199.99.99.*)
copy_to IP_ADDRESS_NSX 35399 break
icmp_unreachable fail;
ip_sa mask 0xFFFFFF00 in (199.99.100.*)
copy_to IP_ADDRESS_NSX 35399 break
icmp_unreachable fail;
ip_sa mask 0xFFFFFFFF in (199.99.101.1)
copy_to IP_ADDRESS_NSX 35399 break
icmp_unreachable fail;
end
```

For the filter attached to the trusted network , make the following changes:

```
filter incom1_ip_spoof_fail
not ip_sa mask 0xFFFFFF00 in (199.99.99.*)
not ip_sa mask 0xFFFFFF00 in (199.99.100.*)
copy_to IP_ADDRESS_NSX 35399 break
icmp_unreachable fail;
end
```

The next change you need to make is based on your security policy. Every allowed service should have a corresponding filter (for example, you should have a filter similar to `incom2_smtp_fail` for e-mail). You need to edit each of these filters in accordance with your policy, which typically is to allow or block network access. For each allowed service in the security policy, edit the current filter with its name. There are two basic cases for allowable services:

- **The first and simplest case is when you want allow all services of this type through.** This is most often the case for the interface attached to a trusted network, because users want unrestricted access to e-mail and WWW services. For example, to allow all outgoing mail traffic, simply comment out the body of the filter *incom(X)_smtp_fail* as follows:

```
filter incom2_smtp_fail

#tcp_dp in (25)

#copy_to IP_ADDRESS_NSX 35399

#icmp_unreachable fail;

end
```

  This allows all outgoing connections to port 25.

- **The second case for allowing a given service involves restricting that service to a specific destination or source IP address.** This type of restriction is usually applied to the interface attached to an untrusted network. In this case, the filter should be edited as follows:

```
filter incom1_icmp_fail

ip_protocol in (1)

#This is where you should enter the typefields of the ICMP

#services you want to block. In this example, we are blocking

#"echo requests" type field 8. To block "timestamp requests"

#you would add 13. tl_byte 0 in (8,13)

tl_byte 0 in (8)

not ip_sa in (198.98.98.*)

not ip_da in (IP_ADDRESS1, IP_ADDRESS2, IP_ADDRESS3, ETC.)

copy_to IP_ADDRESS_NSX 35399

icmp_unreachable fail;

end
```

The above example allows all ICMP echo requests that are from the network 198.98.98.0 or that are destined for IP_ADDRESS1, 2, or 3. All of the filters should comply with your security policy requirements. Please refer to the *NetSentry User's Guide* for more information on this topic.

last.fil—Although this filter is not required, it provides a convenient way to permanently block a particular site. This type of filter might look like this:

```
filter last_block_fail
ip_sa in (BLOCKED_IP_NETWORK)
copy_to IP_ADDRESS_NSX 35399
icmp_unreachable fail;
end
```

Appendix C DBMS Requirements and Setup

In order for nr.sapd to stage data, you will need access to a database management system (DBMS). Oracle is the default database for the SAP package. Data can also be exported to Remedy's ARS system. If you do not have Oracle or Remedy ARS, you can customize *sapd* to support non-Oracle databases, such as Sybase and Informix.

Oracle Install and Setup

Oracle activation for nr.sapd consists of the following steps:

1. **Plan your Oracle environment**
2. **Install/Mount the Oracle product**
3. **Set up environment variables**
4. **Verify generic connectivity**
5. **Create user**
6. **Verify NetRanger–Oracle connectivity**
7. **Create tables**
8. **Setup for Remedy ARS (Optional)**

Plan your Oracle Environment

Ask yourself the following questions about your environment:

- Do you have a local or remote database?
- How much disk space is allocated to the database?
- What tablespaces will be used for NetRanger data?

The answers to these questions will allow you to better install/mount Oracle.

Install/Mount the Oracle Product

You have two setup options for the Oracle Product: local or remote. WheelGroup strongly recommends that you export event data to a remote database. Staging data to a database running locally on the Director system could compromise existing monitoring functionality.

Whether you use a local or remote database, you will need to enter passwords for the Oracle users "sys" and "system" as part of the install process. Remember these passwords because you will use them when setting up the Oracle NetRanger user.

NOTE
You will probably have to increase shared memory and semaphore parameters in /etc/system.

Local

Use the Oracle product CDROM and install with `orainst`. You will need Oracle Server (RDBMS) and SQL*Plus. Make sure to note the directory location of the install because you will use this in setting `$ORACLE_HOME`. Also note the SID, because you will use this to set `$ORACLE_SID`.

Remote

With the client/server model, Oracle RDBMS runs on a server and the client obtains access to it through proprietary DBMS network facilities, such as Oracle's SQL*NET. If you choose a remote setup for Oracle, you will need to choose one of the following setups:

- Remote homogeneous
- Remote heterogeneous

Remote homogeneous
If your client is the same operating system as the server, you can share the Oracle directory from the server and mount it to the client using normal NFS procedures.

Remote heterogeneous
If your client has a different operating system than the server, use the Oracle Installer with the product CDROM to install SQL*Plus and the TCP/IP Protocol adapter. Note the directory location of the install, because you will use this when setting `$ORACLE_HOME`. When the product binaries are available, you will need to configure $ORACLE_HOME/network/admin/tnsnames.ora.

The tnsnames.ora file maps a label (such as "remoteDB") to the access protocols required to establish a connection to that database instance. A sample tnsnames.ora follows:

```
remoteDB =
   (DESCRIPTION=
      (LOCAL=NO)
      (ADDRESS=
         (PROTOCOL=TCP)
         (HOST=DBserver)
         (PORT=1521))
      (CONNECT_DATA=(SID=nr)))
```

Where "remoteDB" = is the label used in the logon string, i.e., username/password@remoteDB. remoteDB is then expanded by SQL*NET into its underlying components, which are as follows:

- HOST=Dbserver
- Port=1521
- Protocol=TCP
- SID=nr

Ensure that the tnsnames.ora file is correctly located and configured; otherwise, you will get an Oracle TNS error message. Oracle will look for it in the following filepaths:

/var/opt/oracle/tnsnames.ora

$ORACLE_HOME/network/admin/tnsnames.ora

$HOME/.tnsnames.ora

UNIX Setup of Oracle Environment Variables

Oracle applications require a number of different environment variables whose settings depend on the setup you have chosen. Refer to the appropriate section for setting environment variables.

Remote or Local

Whether you are using a remote or local setup, Oracle requires the following three basic environment variables under UNIX: `ORACLE_HOME`, `PATH`, and `LD_LIBRARY_PATH`.

The following environment variables should be added to the .profile for the "netrangr" user account; and must contain the following information:

- `$ORACLE_HOME` is the base path for the Oracle product, such as /opt/app/oracle/product/7.1.6 or /usr/apps/oracle/app/product/7.3.2. The path usually ends in the version number of your Oracle product.
- `PATH` must contain `$ORACLE_HOME/bin` to allow access to the Oracle executables.
- `LD_LIBRARY_PATH` must contain `$ORACLE_HOME/lib` to allow access to the runtime libraries.

Local Only

If using a local setup, Oracle requires another environment variable, `ORACLE_SID`, which specifies what data area to use with a local database. When this is the case, update `ORACLE_SID` in the .profile as above.

tty problems (HP-UX)

On certain platforms, such as HP-UX, the default tty settings use the character @ for some built-in commands. You will have a problem connecting to a remote database because the tty settings will delete a portion of your connect string username/password@remoteDB.

To examine the tty setting, execute the following command:

```
% stty -a
```

This will show the mappings for commands. If one of the commands, such as kill, is mapped to @, you will need to change it by executing the following command:

```
% stty kill ^Z
```

Verify Generic Connectivity

Verify that Oracle access is enabled.

Before attempting an Oracle connection, ensure that `$ORACLE_HOME`, `$PATH`, and `$LD_LIBRARY_PATH` are set correctly (as described above).

Local

Set `$ORACLE_SID`. The Oracle daemons must be running. Use the Oracle executable `sqlplus` to verify basic connectivity to Oracle with the following command:

```
% sqlplus sys/password
```

If successful, you will enter into interactive SQL. You will see a prompt similar to the following:

```
Connected to:
Oracle7 Server Release 7.3.2.1.0 - Production Release
PL/SQL Release 2.3.2.0.0 - Production
SQL>
```

NOTE

Ideally, you should omit the password string in the command line call. `sqlplus` will then prompt you for the password. This prevents the password from showing up in output from system applications such as `ps`.

Remote

Use the Oracle utility `tnsping` to verify that the basic network setup is correct (this is similar to the standard ping utility) with the following command:

```
% tnsping remoteDB
```

Use `sqlplus` to verify remote connectivity by appending a valid remoteDB string onto the end of the connection string:

```
% sqlplus sys/password@remoteDB
```

Refer to *Appendix A* for information about the common error messages you might encounter.

Create an Oracle User

Follow these steps before you create a user:

1. Log into Oracle as sys or system (using `sqlplus` or `sqldba`).
2. Determine what Oracle data areas (tablespace) will be assigned for NetRanger usage.

NOTE

This tablespace needs to be as large as possible. The option of assigning a default tablespace to a user eliminates the problem of having to direct subsequent commands to use a certain tablespace. For high volume NSX sensors, you may need to dedicate a TEMPORARY TABLESPACE and a larger ROLLBACK area to the NetRanger user. This is also done in the create user command.

You are now ready to create a user. Follow these steps to create a user:

1. Create the user with the following command:

```
SQL>  create user NetRanger identified by PASSWORD default tablespace 'USERS'
```

   Where PASSWORD is your secret password and USERS is your designated tablespace.

1. Activate the account with the following command:

```
SQL>  grant dba to NetRanger
```

   (Or other lesser privilege based on your site's security policy; at a minimum, the NetRanger user will need privilege to CONNECT, SELECT, INSERT, and DELETE.)

IMPORTANT NOTE

Make sure you update the sapd.conf tokens DBUser and DBPass with the new user and password. *sapd* should not be activated until these are set (unless you are customizing the scripts). Use nrConfigure to change these tokens.

Verify NetRanger–Oracle Connectivity

Verify that the UNIX netrangr user can access the Oracle NetRanger account by running `sqlplus` from the Director platform.

Create Tables

Once you have verified Oracle connectivity and the user netrangr, you can proceed with the creation of the NetRanger database tables. These database tables must be created before *sapd* can start loading the data. Begin by connecting to Oracle as the user netrangr via `sqlplus`.

Once connected, execute the following command at the `SQL>` prompt:

```
SQL> @/usr/nr/bin/sap/skel/create_nr_tables
```

NOTE

The first time you use this procedure, you will receive error messages that say that the tables do not exist. This is normal on the first install.

Be careful when using these create scripts, because they will drop the existing table, thus deleting all the data.

Setup for Remedy ARS (Optional)

Use schema provided in /usr/nr/bin/sap/skel/nr.alarm.schema.def. See the *Using sapx with Remedy* section for more information about Remedy ARS.

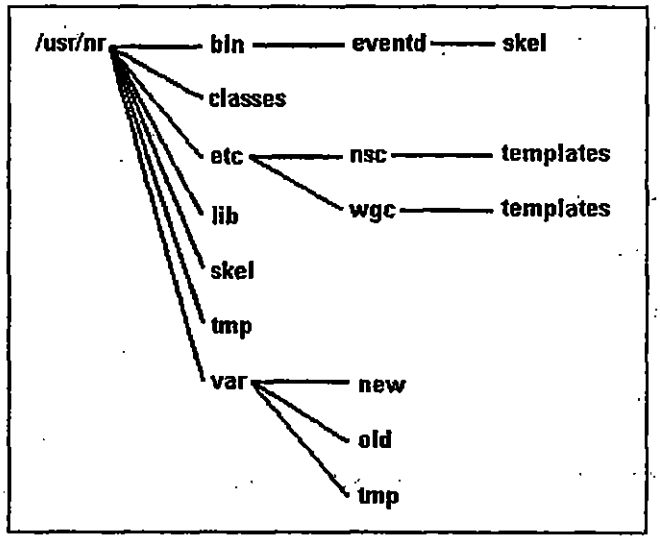
Non-Oracle Install and Setup

Please refer to your DBMS software documentation for installation and setup. Use the *Oracle Install and Setup* section as a template.

Appendix D The NSX File Structure

One of the quickest ways to gain a detailed understanding of NetRanger is to review the NSX file structure and underlying elements. A thorough knowledge of the NSX file structure will help you configure and install the NSX as well as help you better utilize NetRanger's features. It should be noted that this file structure serves as the foundation for both the Director and NSX systems.

The default directory location for the NSX subsystem is /usr/nr, which contains the following directory structure:



This appendix provides a brief explanation of each of these directories.

/bin

This directory contains all of the executables required to run and administer the application services that support an NSX or Director system. These can be loosely grouped into three basic categories:

- Daemon Applications
- Configuration Commands
- System Commands

Daemon Applications

As the *Overview* states, NetRanger is a collection of discrete subsystems that have been implemented via daemons, each with its own well-defined set of services.

NetRanger is also a highly configurable distributed application. The daemon services running on a NSX or Director host depends on the configuration of the application as a whole. By default, a NSX host must have *sensord*, *managed*, *loggerd*, and *postofficed* daemons running. A Director system should have *postofficed*, *smid*, *loggerd*, and *configd* running. In a multi-tiered configuration, one Director system could be configured to only stage data to an RDBMS via *loggerd and sapd*, while another Director system could be dedicated to monitoring and response via the *smid* and *configd* daemons.

The full complement of NSX daemons is as follows:

- nr.configd
- nr.eventd
- nr.loggerd
- nr.managed
- nr.postofficed
- nr.sapd
- nr.smid
- nr.sensord

Each of these daemon services is briefly described below.

configd

This daemon reads and writes data to the NSX configuration files (which are currently restricted to **~/etc**). All communication with this daemon is controlled by the following set of *SNMP-like* executables:

- nrget & nrgetbulk
- nrset & nrunset
- nrexec

NOTE
• This daemon provides *an interface* to configuration information. Actual reconfiguration is performed by *managed*.
• Both command line and graphical user interfaces are provided.

eventd

This daemon is designed to run on Director systems. It allows the Director system to generate user-defined actions for events received by *smid*. A common action is to generate pager notifications via e-mail for alarms of severity "4" and above.

loggerd

This daemon writes out sensor and error data to flat files generated by one or more of the other daemon services. This data is passed to *loggerd* via *postofficed*. *loggerd* creates two basic types of flat files: a single NSX Event file, and one or more IP Session logs. As noted in the *Overview*, data is written to flat files for reasons of performance and fault tolerance. This data can be staged onto a RDBMS via *sapd*.

managed

Whereas *sensord* is responsible for interpreting packet filter events, *managed* is responsible for managing and monitoring these devices. For example, when *sensord* identifies that a certain type of attack should be shunned, it sends a shun command to *managed* via the post office facility. *managed* then reconfigures the packet filter appropriately. Similar commands can be sent to this daemon from the Director. *managed* can also be polled for operational statistics such as:

- the number of network devices being managed
- device type
- device statistics (packets/sec, bytes/sec, etc.)

postofficed

This daemon serves as the communication vehicle for the entire NetRanger product. All communication between daemons is routed through this service. In most NetRanger configurations the NSX and Director systems reside on separate hosts. Each system relies on a *postofficed* to maintain communication with the other. Note that a *postofficed* is required even when all of the systems are located on a single host.

This daemon service include the following features:

- A proprietary connection-based protocol that resends a message whenever any of its packets are lost.
- Point-to-point routes that may include intermediate post office nodes.
- Communication integrity that is maintained via alternate routes (specified in ~/etc/routes and ~/etc/destinations). When one route falls, communication is switched to an alternate route. Communication is reestablished to the preferred route as soon as it comes back online.

Routing is based on a three part key that includes the following tuples:

- Organization
- Host
- Application

The Organization IDs are defined in ~/etc/organizations; Host IDs are defined in ~/etc/hosts; Application IDs are defined in ~/etc/services.

sapd

This daemon is a light weight scheduler that pulls data from event log files and exports it into either an Oracle or Remedy database via DBLOAD processes such as sapx. *Sapd* also can process IP Session logs via ARCHIVE processes.

smid

This daemon routes messages to and from the Director and other daemon services, such as *sensord.* This service relies on routing information stored in the Director's ~/etc/hosts file. In addition to being able to communicate with other daemons, *smid* can redirect messages to other daemon services, such as eventd and loggerd based on *DupDestination* entries in etc/smid.conf.

sensord

This daemon interprets and responds to all of the events generated by one or more packet filter devices. Data is read in from Network Device sockets and sent to *postofficed* for routing to other daemon services. The types of events received by sensord is a function of the filter policy on each packet filter device. The types of messages *sensord* sends to other daemons is based on information stored in ~/etc/sensord.conf. Where these messages are routed is defined in ~/etc/destinations.

Although *sensord* supports 256 levels of alarm events (0-255), only levels 0-6 are currently being used. Level "0" is an internal alarm that *sensord* uses to track such services as RIP, ICMP, and so on; it is never routed to *postofficed.* Levels 1-6 are sent to *postofficed*, and they identify increasing patterns of misuse, where "5" is the highest type of alarm. Note that every alarm message includes an Alarm-Id that identifies a specific attack signature. These alarm signatures are enumerated in ~/etc/sensord.conf as SigOfGeneral entries.

In addition to generating alarm messages for a Director application, *sensord* can automatically instruct *managed* to shun an attack for a specified period of time.

Configuration Commands

The NSX Sensor system comes with the following built-in commands that allow remote devices to configure and gather information about the NSX. As noted in *Operating the Director*, these commands serve as the foundation for the Director's **nrConfigure** interface.

- nrget
- nrgetbulk
- nrset
- nrunset
- nrexec

nrget

This file retrieves single pieces of information from the NSX.

```
nrget socket appid hostid orgid priority token [ identifier... ]
```

nrgetbulk

This file retrieves multiple pieces of information from the NSX.

```
nrget socket appid hostid orgid priority token [ identifier... ]
```

nrset

This file sets attributes about the NSX.

```
nrset socket appid hostid orgid priority token [ identifier...] value [value ...]
```

nrunset

This file unsets attributes about the NSX.

```
nrset socket appid hostid orgid priority token [ identifier... ] value [value ... ]
```

nrexec

This file executes commands on the NSX.

```
nrset socket appid hostid orgid priority token [ identifier... ] value [value ... ]
```

System Commands

The following scripts are used to perform maintenance on the NSX:

- install
- nrstart
- nrstop
- nrstatus

install

This script is used to install the startup/shutdown files for NetRanger into the UNIX /etc/initd directory. The command takes the following form:

```
install { add | remove }
```

nrstart

This script starts the NSX. It supports the following options:

- -d Don't daemonize
- -h Show help
- -v Show version number

The command takes the following form:

```
nrstart [-d] [-h] [-v]
```

nrstop

Executing this script stops the NSX.

nrstatus

Executing this script returns the currently running daemons.

/classes

This directory serves as the *rooted class path* for the Director's Java applications, such as **nrConfigure.**

/etc

This directory contains two types of configuration files: **daemon-specific** and **NSX system** files. All of NetRanger's configuration files are currently ASCII flat files.

Daemon Configuration Files

There is a one-to-one correspondence between daemons and their configuration files. Whereas the naming convention for a daemon is nr.<daemon>, the convention for its configuration file is <daemon>.conf. Each of these files contains token-based configuration information of the form: <token> <value>. For example, the error file for *nr.managed* is identified in *managed.conf* as follows:

```
FilenameOfError    ../var/errors.managed
```

Although these entries can be set via a text editor, all changes should be managed via **nrConfigure**. Otherwise, the chance of generating errors increases.

- *configd.conf*
- *eventd.conf*
- *loggerd.conf*
- *managed.conf*
- *postofficed.conf*
- *sapd.conf*
- *sensord.conf*
- *smid.conf*

With the exception of *managed.conf*, *sensord.conf*, and *smid.conf* these files currently contain little more than timing intervals and the name of the ~/var/error file for each daemon service. The remainder of this section is devoted to explanations of *managed-*, *sensord-* and *smid.conf*.

managed.conf

This file is used by the NetRanger NSX to configure the management daemon that controls actions on the router. Two types of tokens in this file require special attention: *NetDevice* and *NeverShunAddress*.

NetDevice identifies the IP address and the system password of an NSG BorderGuard the NSX Sensor is connected to. A separate *NetDevice* entry is required for every BorderGuard system being managed by the NSX system. The format of this token is as follows:

```
NetDevice    ROUTER_IP_ADDRESS       NSG_Borderguard_v1   ROUTER_PASSWORD
```

NeverShunAddress identifies the IP address of a host or network device you never want the NSX to BorderGuard to shun. At the very minimum you make sure that this file contains *NeverShunAddress* entries for the local system and the remote NSX or Director system. Other addresses can be added based on your operational needs. Example entries might look as follows, where <255.255.255.255> identifies the subnet mask for the IP address you do not want shunned.

```
NeverShunAddress ROUTER_IP_ADDRESS       255.255.255.255
NeverShunAddress NSX_IP_ADDRESS          255.255.255.255
NeverShunAddress INTERNAL_NETWORK        255.255.255.255
```

NOTE

NeverShunAddress entries do not bypass existing filters, but rather prevent them from being completely blocked out of the router.

sensord.conf

This file serves as the heart of the NSX Sensor system, and is the largest of all of the configuration files. It can be logically broken down into the following sections:

- General
- Event Specification
- Strings To Look For
- Matched String and Event Levels
- TCP/UDP Ports and Event Levels
- Policy Violations
- Excluded Events

General
At a minimum, you will need to edit the following lines:

```
RecordOfInternalAddress  INTERNAL_NETWORK_ADDRESS INTERNAL_NETWORK_NETMASK
RecordOfInternalAddress  ROUTER_EXTERNAL_IP_ADDR  255.255.255.255
```

These two lines establish which networks NetRanger will protect. **It is important to include all internal addresses because this information is also used for intrusion detection.** You should have one entry for each network address and one for the router external interfaces.

```
RecordOfDataSource      ROUTER_IP_ADDRESS    255.255.255.255
```

This line determines which BorderGuard the sensor daemon will receive information from. You need to fill in the router IP address for this line.

```
RecordOfLogAddress    LOGGED_NETWORK_ADDRESS    255.255.255.0
```

This line establishes a binary log of a particular host or network. This is most commonly used to support an investigation and creates a binary record of all data originating from a particular source address. Initially, this line will probably be commented out.

```
MinutesOfAutoLog        LOG_MINUTES
MinutesOfAutoShun       SHUN_MINUTES
```

These two lines establish the amount of time that NetRanger will shun or log after a particular alarm is received.

Event Specification
The first task-specific segment is "Event Specification", which is based upon the SigOfGeneral token. This token identifies an *action* and *alarm levels* for various types of events. *sensord.conf* contains a number of different `SigOf-` tokens, all of which share the same basic format, which is defined as follows:

```
SigOfGeneral  SIGID  ACT   D1   D2   D3   D4   # Description of event
```

NOTE

`SIGID` is a numeric identifier that establishes a logical link to internal signature ids in the *sensord* daemon, and should be treated as **read-only** information. Any changes or additions to these entries will disrupt NSX functionality.

The first column that can be configured is the `ACT` column. It contains a number that specifies how to respond (an *action*) to a particular event. Valid `ACT` settings include:

0 - no action

1 - shun

2 - log

3 - shun and log

The remainder of SigOfGeneral maps alarm levels to each of the destinations specified in /usr/nr/etc/destinations. This makes it possible to uniquely define the severity of an event for every destination the NSX is configured to talk to.

The following SigOfGeneral defines what to do when a `TCP Port Sweep` is encountered. It first states that no automatic action should be taken (`ACT=0`). It also defines that level "5" alarms should be sent to its first three destinations, and a level "4" alarm to the last destination.

```
SigOfGeneral  3001   0   5   5   5   5   # TCP port sweep
```

Strings To Look For

This section defines the strings to look for within a given TCP/IP service. The general format for an entry in this section is the following:

```
RecordOfStringName  StringID  TCP/IP Port  Direction  Num Occurances  "Matched String"
```

`StringID` establishes the unique ID for that particular string. `TCP/IP port` tells NetRanger which port to watch to look for this string. Note that *first.fil* must have a *Copy_To* entry for this port that copies all of its packets to the NetRanger NSX. `Direction` tells the NSX which direction to look for this string::

1 - external-to-internal

2 - internal-to external

3 - both directions

The number of occurrences indicates how many times the NSX must see this string before it alarms. `Matched String` is the actual string to be matched. The exact format is controlled by regular expression (regex) syntax.

Matched String and Event Levels

This section identifies action and alarm levels for the string matches defined in the previous section. `SigOfStringMatch` uses the same format as `SigOfGeneral` described above, and is linked to a `RecordOfStringName` by its `StringID`.

```
SigOfStringMatch  StringID  ACT  D1  D2  D3  D4  # Description of event
```

TCP/UDP Ports and Event Levels

This section establishes the alarm levels for any packet, successful or not, with a TCP or UDP destination port corresponding to the entry. The action fields and the event level specification are the same as in previous sections.

The following entry illustrates that all Telnet attempts (Port 23) should have no action taken but should generate a level 2 alarm:

```
        <port> <action> <dest1> <dest2> <dest3> <dest4>
SigOfTcpPacket    23      0       2       2       2       2
```

The next entry indicates level 3 alarms for all TFTP (port 69) attempts which logs the packets:

```
        <port> <action> <dest1> <dest2> <dest3> <dest4>
SigOfUdpPacket    69      1       3       3       3       3
```

Policy Violations

This section identifies the filter names and IDs for all BorderGuard filters that can pass information to the NSX. In the following example, `Income_1_Filter_Fail` has a filter ID of `1000`.

```
      RecordOfFilterName  1000   Income_1_Filter_Fail
```

Each of these policy filters has a matching SigOfFilterName that identifies its action and destinations. The format is identical to the SigOf- tokens described above. In the following example a `Income_1_Filter_Fail` event will be logged to the second destination defined in /usr/nr/etc/destinations.

```
        <filter_id> <action> <dest1> <dest2> <dest3> <dest4>
SigOfFilterName   1000      2       2       2       2       2
```

Excluded Events

The final section of */usr/nr/etc/sensord.conf* is the "Excluded Events" segment. This is where alarms can be *locked out* for events at specific source addresses. The standard format follows:

```
     RecordOfExcludedAddress  SigID  SubSigID   IP Address
```

`SigID` corresponds to the primary signature established in the "Event Specification" section that establishes all of the primary IDs. `SubSigID` corresponds to those signatures established in the latter sections, such as those for filtering and string matching.

smid.conf

This file is the configuration file for *sapd*, which passes information to the Director *nrdirmap* application for display within the OpenView user interface. The only token that needs to be edited in this file is `DupDestination`, which defines duplicate destinations for the events. This token allows a Director system to forward event information onto other services and systems. In the following example, events, commands, and errors of level "1" and above are forwarded onto *loggerd*. This allows the Director to monitor ***and*** log events based on a single NSX data stream. This in turn, helps reduce the amount of network traffic a NSX must transmit to a Director. Instead of having to transmit two identical (but separate) data streams to `director1.wheelgroup`, the NSX is able to send a single stream that is duplicated on the Director platform.

```
DupDestination  director1.wheelgroup  loggerd  1  ERRORS,COMMANDS,EVENTS
```

As the next example shows, this token can also be used to forward events onto another Director system. In this case, however, only events of level "2" and above are forwarded.

```
DupDestination  director2.wheelgroup  smid  2  ERRORS,COMMANDS,EVENTS,IPLOGS
```

NOTE

The host and organization IDs must match entries defined in /usr/nr/etc/hosts.

NSX System Files

The */usr/nr/etc* directory contains all of the NSX's system files. These files are modeled after Unix */etc* files and Unix system administrators should have no problem understanding their format and entity relationships. The files currently are as follows:

- auths
- daemons
- destinations
- hosts
- organizations
- routes
- services
- signatures

NOTE

These files are *described* in order of use rather than alphabetically.

organizations

This file identifies all of the *organizations* and their *organization Ids* currently registered for a given NetRanger system. The format of the file is:

```
<org_id> <organization_name>
```

A typical file might look as follows:

```
100   wheelgroup
101   netsolve
102   btg
1001  dataworks
```

NOTE

This file serves as a *global* registry that must agree across all of the NSX and Director systems within a NetRanger domain.

hosts

This file is much like a unix /etc/hosts file. It enumerates the organizations and hosts that are recognized by a given NSX or Director system in a NetRanger configuration. Each entry has the form:

```
<host_id>.<organization_id>   <host_name>
```

where *host_id* and *organization_id* are numeric identifiers assigned by WGC, and *host_name* is a DNS-like name assigned by the user. In addition to a list of recognized hosts, this file always contains a *localhost* entry.

A typical hosts file might look like the following:

```
10.100   localhost
10.100   admin.wheelgroup
11.100   dev.wheelgroup
12.100   data.wheelgroup
13.100   nsx-1.wheelgroup
14.100   nsx-2.wheelgroup
15.100 nsx-3.wheelgroup
```

NOTE

Entries for each host must be consistent across *all* NSX and Director systems. Otherwise, some systems will generate "unrecognizable host" messages in ~/var/errors .postofficed.

services

This file defines the *application_id* for each of the daemon services found in **~/bin**. When combined with the *host_id* and *organization_id*, these identifiers uniquely identify a daemon within a NetRanger security map. Each entry has this form:

```
<application id>  <daemon name>  [<comment>]
```

All NSX hosts should have the same default services file, which is currently defined as follows:

```
10000 postofficed # Provides the NetRanger comm system
10001 sensord  # Monitors network traffic for security
10002 configd  # Sets and Gets configuration information
10003 managed  # Manages NetRanger components
10004 eventd   # Sends messages and alarms to pagers
10005 loggerd  # Logs events, cmds, errors, IP logs, etc.
10006 smid     # Routes msgs & events to nrdirmap
10007 sapd     # Stages log data into a DBMS
```

auths

This file identifies which type of *configd* operations are authorized for the each of the hosts listed in this file. Each entry follows this form:

```
<host name> <configd services>
```

In the following example, `admin.wheelgroup` has full authorization, whereas `data.wheelgroup` can only read configuration information.

```
admin.wheelgroup      GET,GETBULK,SET,UNSET,EXEC
nsx-1.wheelgroup      GET,GETBULK
nsx-2.wheelgroup      GET,SET,UNSET,EXEC
```

NOTE

Each entry must have a corresponding entry in ~/etc/hosts.

destinations

NetRanger is a distributed application that has the ability to route messages from a given post office to any of the daemon services registered in the ~/etc/hosts and ~/etc/services files. The destinations file identifies what type of messages get routed to a given application on a given host. Each entry is of the following form:

```
<Destination Id.> <host name> <daemon name> <message level> <message type>
```

Where `<Destination Id>` is a Numeric Id. Up to 32 destinations are currently allowed. `<host name>` must be an entry from ~/etc/hosts. `<daemon name>` must be an entry from ~/etc/services. `<message level>` identifies the level of messages that will be routed. Messages below that level are not passed on to *postofficed*. `<message type>` identifies the types of messages to route.

For example, in the following destinations file, level 2 messages of the type *event, error, command,* and *IP_log* are being routed to the *smid* daemon on the host *admin.wheelgroup*. This type of configuration would normally be found on an NSX system, where *admin.wheelgroup* is a Director system.

```
1   admin.wheelgroup   smid   2 EVENTS,ERRORS,COMMANDS,IPLOGS
```

routes

This file identifies the actual IP routes that *postofficed* uses to send messages between different hosts. Each entry is of the following form:

```
<host name> <connection #> <IP address> <UDP port> <Type>
```

`<host name>` must be an entry from ~/etc/hosts. `<connection #>` identifies the priority of this entry relative to the other routes enumerated in this file. The lower the number, the higher the route priority. If there is more than one entry of the same priority, *postofficed* accepts the last entry of that priority. `<IP address>` identifies the **end-point** IP address of the **remote system.** `<UDP port>` identifies the UDP port service to route through on each host. These ports are usually in the 30-50000 range. `<Type>` identifies whether the connection is a dial-up vs. dedicated connection. This field is currently not used.

In the following example, `nsx-2.wheelgroup` serves as the primary route to the IP Address 10.1.7.12, and `nsx-3.wheelgroup` is the secondary route to the same host.

```
nsx-1.wheelgroup     1  10.1.7.9 45000   1
nsx-2.wheelgroup     1  10.1.7.12   45000    1
nsx-3.wheelgroup     2  10.1.7.12   45000    1
```

daemons

This file contains the names of daemons that should be started when the system starts up. Each entry is of the form:

```
<daemon name>
```

For example, the following sample entry would start *postofficed*, *configd*, *loggerd*, and *smid* on a Director system:

```
nr.postofficed
nr.configd
nr.loggerd
nr.smid
```

NOTE

Whereas the *services* file identifies all of the daemon services a given NSX or Director system is *capable* of running, the *daemons* file identifies the services to launch upon startup.

signatures

This file associates *signature ids* with *signature names* for NetRanger applications, and is used primarily by Director systems. The format of each entry is in the file is:

```
<signature id> "<signature name>"
```

For example, the following sample entry defines a few attack signatures.

```
1000 "Bad Option List"
1001 "Record Packet Rte"
1002 "Timestamp"
1003 "Provide s,c,h,tcc"
1004 "Loose Src Rte"
1005 "SATNET ID"
```

NOTE

This file should NOT be modified.

~/skel

This directory contains basic UNIX configuration files (such as .profile) that are required to configure remote login accounts.

~/tmp

This is the directory where sockets are created by the different daemons. The names of these sockets are dictated by the parent daemon. *postofficed* creates sockets with names such as mailbox.sensord and mailbox.configd. *configd* creates sockets with names such as socket.command.

~/var

This is the default location for all of the log files generated by *loggerd* and error files for all of the applications listed in the ~/etc/daemons file.

Log Files

Two different types of log files are created by *loggerd:* **Event** and **IP Session** logs.

Event File

Although the name of this file can be changed via nrset, the default name of this file is currently hard-coded in *loggerd*. The name of event logs are based on the date and time a new log began (e.g., log.199607291332). Although even message types are defined, only **three** are currently implemented.

Message Type	Description
0	Default
1	Command
2	Error (Defined)
3	Command Log (Defined)
4	Event (Defined)
5	IP Log
6	Redirect

The schema for defined message types are as follows:

Error records from *managed* (ERROR)

```
1, MsgType (SInt) nrPROTOCOL_ERROR,
2, ULong RecordID,
3,4,5 timestamp ( ULong time, String year/month/day,hour:min:sec )
6, ULong addr.ApplID,
7, ULong addr.HostID,
8, ULong addr.OrgID,
9, String ErrorData
```

Command Log records from *configd* (COMMANDLOG)

```
1, MsgType (SInt) nrPROTOCOL_COMMANDLOG,
2, ULong RecordID,
3,4,5 timestamp ( ULong time, String year/month/day,hour:min:sec )
6, ULong addr.ApplID,
7, ULong addr.HostID,
8, ULong addr.OrgID,
9, ULong src.ApplID,
10, ULong src.HostID,
11, ULong src.OrgID,
12, String LogData
```

Event/Alarm records from *sensord* (PROTOCOL_EVENT)

```
1, MsgType (SInt) nrPROTOCOL_EVENT,
2, ULong RecordID,
3,4,5 timestamp ( ULong time, String year/month/day,hour:min:sec )
3,4,5 timestamp ( ULong time, String year/month/day,hour:min:sec )
6, ULong addr.ApplID,
7, ULong addr.HostID,
8, ULong addr.OrgID,
9, String from,
10, String to,
11, SInt event->level,
12, ULong event->SigID,
13, ULong event->SubSigID,
14, String protocol,
15, String sourceIpAddress,
16, String destIpAddress,
17, SInt event->SrcPort,
18, SInt event->DstPort,
19, String routerIpAddress,
20, String EventData
```


Session Log Files

In addition to detecting attack signatures, *sensord* is able to monitor the UDP traffic associated with a specific type of attack. For example, *sensord* can be configured to monitor all of the packets associated with an IP spoof. *loggerd* creates a separate log file in **~/var** for each of these monitoring sessions. The name of each session log file is based on the IP address of the attacking host (e.g., log.209.15.163.54).

Appendix E NetRanger Hardware Components

NSX Sensor Hardware Components

- Rack-Mounted PC w/Solaris x86 2.5
- 166 MHz Pentium® processor (10Mbps)
- 110 MHz UltraSPARC® (100Mbps)
  - ◊ wide SCSI
- 32 MB RAM
- Three NSX Versions:
  - ◊ NSX 1000
  - ◊ NSX 2000
  - ◊ NSX 5000
- Four configuration options:
  - ◊ access through serial port
  - ◊ network access
  - ◊ modem dial-up
  - ◊ video/keyboard

NSX Versions

- ***NSX 1000***
  - ◊ Pentium 166 NSX
  - ◊ NSG BorderGuard 1000
    - ⇒ 2 10BaseT / AUI Ethernet
    - ⇒ 3 WAN Daughtercards
    - ⇒ ISDN
  - ◊ Up to 512 Kbps

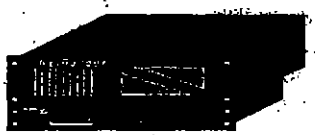


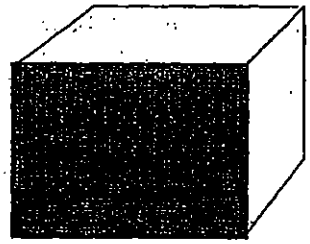
Figure D-1: The NSX 1000

- ***NSX 2000***
  - ◊ Pentium 166 NSX
  - ◊ NSG BorderGuard 2000
    - ⇒ 4 Ethernet
    - ⇒ 4 WAN
    - ⇒ 2 Ethernet/ 2 WAN
  - ◊ 512 Kbps to 10 Mbps Ethernet

Figure D-2: The NSX 2000

- ***NSX 5000***
  - ◊ UltraSPARC NSX
  - ◊ NSG ERS/Passport
    - ⇒ FDDI
    - ⇒ Ethernet
    - ⇒ WAN (T1 and T3)
    - ⇒ (ATM)
  - ◊ 10 to 100 Mbps Ethernet

Figure D-3: The NSX 5000



NSG BorderGuard Hardware Components

- The BorderGuard is a bridge/router that links IEEE 802.3/Ethernet networks, and other networks via WAN links (e.g., ISDN, leased lines).
- Includes Packet Control Facility (PCF) for firewall routing
- Includes Bridge Control Facility (BCF) for filtered bridging
- Routing support
  - ◊ TCP/IP
  - ◊ DECnet Phase IV
  - ◊ Novell IPX
  - ◊ Banyan VINES
  - ◊ AppleTalk II
  - ◊ XNS

Appendix F Uninstalling the Director

Uninstallation Process for HP-UX Systems

This section describes how to remove the Director software from an HP-UX system. Please note that it does not describe how to remove HP OpenView—OpenView will still be installed after executing the steps in this section. If you want to remove OpenView, uninstall the Director first, then refer to the HP OpenView documentation for the uninstallation procedures.

NOTE

Note: If you want to uninstall the Director but continue to use HP OpenView, you might consider deleting your NetRanger/Director data from the OpenView databases *before* uninstalling the Director. **Once you uninstall the Director, there will be no way to remove the data from the databases, other than by completely removing the databases.** To delete data from the databases while the Director is still installed, use the **Delete Object** menu item from the user interface.

To uninstall the part of the NetRanger/Director that integrates with OpenView, follow these steps:

1. **Instruct all users to exit the OpenView user interface by choosing the Map→Exit menu function.**
2. **Logon as user netrangr and stop the NetRanger daemons using this command:**

   ```
   /usr/nr/bin/nrstop
   ```
3. **Logoff as user netrangr and logon as user root.**
4. **Instruct all users who are logged on as user netrangr to log off now.**
5. **To remove the Network Management Interface software, type:**

   ```
   /usr/sbin/swremove -v -x mount_all_filesystems=false director
   ```
6. **To remove the Remote Configuration software, type:**

   ```
   /usr/sbin/swremove -v -x mount_all_filesystems=false WGCcfgs
   ```
7. **To remove the RDBMS software, type:**

   ```
   /usr/sbin/swremove -v -x mount_all_filesystems=false WGCsapd
   ```
8. **To remove the NSX Interface software, type:**

   ```
   /usr/sbin/swremove -v -x mount_all_filesystems=false nsx
   ```

To remove the **netrangr** userId, follow these steps:

1. **On HP systems, bring up the SAM utility by typing the following:**

   sam &

2. **Select** Accounts for Users and Groups.
3. **Double-click the** Users **icon.**
4. **Select** netrangr **from the list.**
5. **From the** Actions **menu, select** Remove.
6. **Determine whether or not you want all of the files owned by** netrangr **to be deleted, and make the appropriate selection in the dialog box.**
7. **Press** OK.
8. **Press** Yes **to remove the user.**
9. **Press** OK **when the process is complete:**

To remove the **netrangr** group (HP-UX only), follow these steps:

1. **Double-click on the** Groups **icon.**
2. **Select** netrangr **from the list.**
3. **From the** Actions **menu, select** Remove.
4. **Determine how files that belong to this group should be treated, and make the appropriate selection in the dialog box.**
5. **Press** OK.
6. **Press** Yes **to remove the group.**
7. **Press** OK.

Uninstallation Process for Sun Solaris Systems

This section describes how to remove the Director software from a Sun Solaris system. Please note that it does not describe how to remove HP OpenView—OpenView will still be installed after executing the steps in this section. If you want to remove OpenView, uninstall the Director first, then refer to the OpenView documentation for the uninstallation procedures.

NOTE

Note: If you want to uninstall the Director but continue to use HP OpenView, you might consider deleting your NetRanger/Director data from the Solaris databases *before* uninstalling the Director. **Once you uninstall the Director, there will be no way to remove the data from the databases, other than by completely removing the databases.** To delete data from the databases while the Director is still installed, use the **Delete Object** menu item from the user interface.

To uninstall the part of the NetRanger Director that integrates with OpenView, follow these steps:

1. **Instruct all users to exit the OpenView user interface by choosing the Map→Exit menu function.**
2. **Logon as user netrangr and stop the NetRanger daemons using this command:**

   ```
   /usr/nr/bin/nrstop
   ```
3. **Logoff as user netrangr and logon as user root.**
4. **Instruct all users who are logged on as user netrangr to log off now.**
5. **Type the following command to remove the Director user interface application:**

   ```
   pkgrm WGCdrctr
   ```
6. **Type the following command to remove the RDBMS software:**

   ```
   pkgrm WGCsapd
   ```
7. **To remove the Remote Configuration software, type this command.**

   ```
   pkgrm WGCcfgs
   ```
8. **Type the following command to remove the NSX interface software:**

   ```
   pkgrm WGCnsx
   ```
9. **Remove the netrangr user id (optional) with the following commands:**

   ```
   userdel -r netrangr
   groupdel netrangr
   ```

Index

E

F

G

H

I

L

M

N

O

P

S

T

U

V

WheelGroup Corporation

WheelGroup Corporation welcomes your ideas on how to improve our documentation. Please take a moment to answer the questions below by checking either the "Yes" or "No" box. Explain any "No" responses in the sections provided. Include other comments in the "Comments" section. When you have finished, please detach and mail this page, or fax it to (210) 494-6303. You may also e-mail us about documentation issues at documentation@wheelgroup.com.

Does this manual contain all the information you expected?	◊ Yes	◊ No
Are all procedures documented in the manual correct?	◊ Yes	◊ No
Is this manual easy to read and understand?	◊ Yes	◊ No
Are examples and diagrams helpful?	◊ Yes	◊ No
Are there enough examples and diagrams?	◊ Yes	◊ No
Other comments:		

CERTIFICATE OF SERVICE

I hereby certify that on the 30th day of June, 2006, I electronically filed the foregoing document, **DECLARATION OF PAUL S. GREWAL IN SUPPORT OF SYMANTEC CORPORATION'S OPPOSITION TO SRI INTERNATIONAL, INC.'S MOTION TO EXCLUDE FROM EVIDENCE THE EXPERT OPINION OF DANIEL TEAL, VOLUME 3 OF 4,** with the Clerk of the Court using CM/ECF which will send notification of such filing to the following:

John F. Horvath, Esq.
Fish & Richardson, P.C.
919 North Market Street, Suite 1100
Wilmington, DE 19801

Richard L. Horwitz, Esq.
David E. Moore, Esq.
Potter Anderson & Corroon LLP
Hercules Plaza
1313 North Market Street, 6th Floor
Wilmington, DE 19801

Additionally, I hereby certify that on the 30th day of June, 2006, the foregoing document was served via email and by Federal Express on the following non-registered participants:

Howard G. Pollack, Esq.
Michael J. Curley, Esq.
Fish & Richardson
500 Arguello Street, Suite 500
Redwood City, CA 94063
650.839.5070

Holmes Hawkins, III, Esq.
King & Spalding
191 Peachtree Street
Atlanta, GA 30303
404.572.4600

Theresa Moehlman, Esq.
King & Spalding LLP
1185 Avenue of the Americas
New York, NY 10036-4003
212.556.2100

/s/ Richard K. Herrmann
Richard K. Herrmann (#405)
Mary B. Matterer (#2696)
Morris, James, Hitchens & Williams LLP
222 Delaware Avenue, 10th Floor
Wilmington, DE 19801
(302) 888-6800
rherrmann@morrisjames.com

Counsel for Symantec Corporation